

# Encoding Quadrilateral Meshes

Asish Mukhopadhyay      Quanbin Jing  
School of Computer Science  
University of Windsor  
Windsor, Ontario

## Abstract

An important problem in geometric compression is to find succinct representations (encoding schemes) for the connectivity of polygonal meshes. In this note, we show that the encoding scheme discussed in [1] for quadrilateral mesh connectivity can be improved from 3.5 bits per vertex to less than 3 bits per vertex. We also show that an easy equivalence between the labelling schemes of King et al [2] and of Kronrod-Gotsman [1], improves this further to 2.67 bits per vertex. The same upper bound has also been reported in [2], making an involved use of the CLRES labelling scheme.

## 1 Introduction

Following the publication of Deering's paper [3], geometric compression has become a very active field of research [2], [1]. The emphasis of the research has primarily been on finding efficient schemes for encoding the geometry (connectivity) of polygonal meshes. The efficiency of such schemes are quantified by the number of bits needed per vertex of the mesh or the numbers bits required per edge of the mesh. The practical significance of this is that it enables one to store and transmit such meshes (over the Internet) succinctly. In this note we show that the encoding scheme discussed in [1] for meshes made up of quadrilaterals only (quad mesh, for short) can be improved to less than 3 bits per vertex. We also show that an easy equivalence between the labelling schemes of King et al [2] and of Kronrod & Gotsman [1], im-

proves this further to 2.67 bits per vertex. The same upper bound has also been reported in [2], making an involved use of the CLRES scheme.

## 2 Kronrod-Gotsman scheme

The Kronrod-Gotsman scheme [1] generalizes the CLRES labelling scheme of [2] to non-triangular meshes. Their main observation is that as we traverse a mesh (with or without boundary) in depth-first order, the interaction of each polygon with the rest of the mesh can be enumerated in a finite number of ways. For example, in a quad mesh each quad interacts with the rest of the mesh in one of 13 ways (Fig.1, arrows indicate the current gate) and hence this interaction can be coded in a unique manner. It is easy to enumerate all these cases if we note that each of the remaining three edges of the current quad either belongs to the mesh boundary or doesn't, and so also for the remaining two vertices.

The compression process traverses the mesh in depth-first order, starting with a quad, at least one of whose edges is part of the mesh boundary. Note that if a mesh is closed we can create a boundary by removing one of the polygons. In the following discussion, the term *gate* will mean an edge of a quad that we are currently visiting, and one that it shares with the current mesh boundary.

For example, if we traverse the quad mesh of Fig.2, starting with the thick edge, and always choose the next gate to be the edge of the current quad that is counterclockwise with respect to the current gate then we get the following output string:

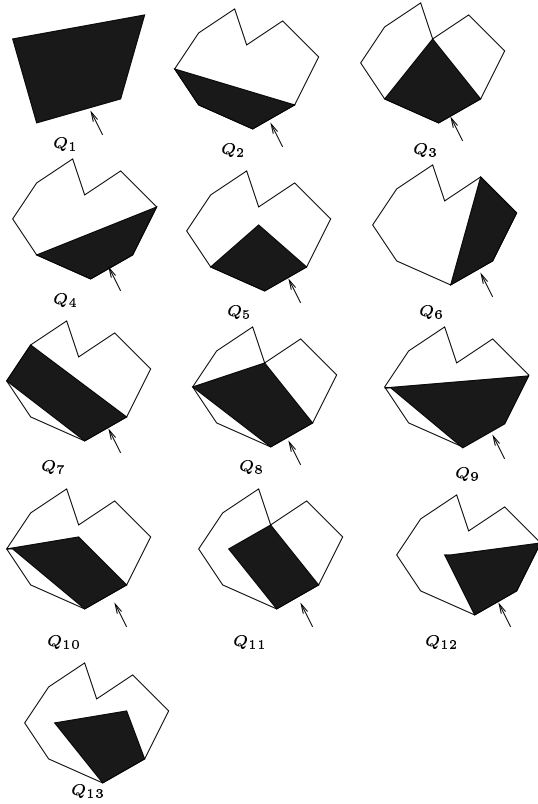


Figure 1: *Interaction of a quad with the mesh*

$Q_{13} Q_6 Q_6 Q_5 Q_{12} Q_{12} Q_6 Q_6 Q_{12} Q_{12} Q_9 Q_1 Q_6 Q_1$

### 3 Encoding scheme with less than 3.0 bpv

Kronrod & Gotsman [1] proposed a prefix-free variable length encoding scheme for such a string that needs at most 3.5 bits per quad. We show that this can be improved to less than 3 bits per vertex.

As we process a quad, we introduce new edges and vertices. These new edges and new vertices are free edges and vertices that become part of the mesh boundary when we process and remove the current quad. An edge or vertex of a quad is *free* if it doesn't belong to the mesh boundary. Table 1 summarizes

Type	# of new edges	# of new vertices
$Q_1$	0	0
$Q_2$	1	0
$Q_3$	2	0
$Q_4$	1	0
$Q_5$	2	1
$Q_6$	1	0
$Q_7$	2	0
$Q_8$	3	0
$Q_9$	2	0
$Q_{10}$	3	1
$Q_{11}$	3	1
$Q_{12}$	2	1
$Q_{13}$	3	2

Table 1: *Mesh-Quad Interactions*

this information for each of the thirteen types of quad that is encountered during mesh traversal.

We also have the following important observation about the encoding process.

**Claim 1** *In a manifold mesh, a quad of type  $Q_5, Q_{10}, Q_{12}$ , or  $Q_{13}$  is never followed by a quad of type  $Q_1, Q_2, Q_3, Q_4, Q_5$ , if while traversing the mesh we choose the next gate to be situated counterclockwise with respect to the present one.*

**Proof:** This is immediate as each quad of type  $Q_5, Q_{10}, Q_{12}$  or  $Q_{13}$  leaves a free vertex and the choice of the next gate makes it impossible for the next quad to have the edge previous to the gate on the active edge boundary as is required to have a quad of type  $Q_1, Q_2, Q_3, Q_4, Q_5$ .

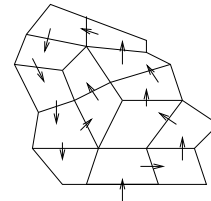


Figure 2: *Traversing a quad mesh*

**Note 1** In the paper by King et al [2], they point out an exception to the above claim when the quad mesh has a an internal valence-two vertex (that is, a vertex on which exactly two quads are incident) and deals with this situation separately, resulting in an encoding scheme that has an upper bound of more than 3 bits per vertex. For a manifold mesh, the claim is true without any exception.

In view of the above observation, we borrow an idea from the coding scheme of [2], to set the code for each interaction-type as in the table below to obtain a variable-length prefix-free coding scheme.

We argue below why the above coding scheme provides an upper bound of 3.0 bits per vertex for the Kronrod-Gotsman scheme.

We assume that we have a quad mesh homeomorphic to a sphere. Let  $E$  be the number of its edges,  $V$  the number of its vertices and  $Q$  the number of quads it has. Since each edge is shared by exactly two quads,  $E = 2Q$ . Combining this with Euler's formula, we get

$$V = Q + 2 \quad (1)$$

For a very large mesh,  $Q \gg 2$ ; therefore, ignoring the additive term in (3) above, we can assume that  $V = Q$ .

Let  $|Q_i|$  denote the number of quads of type  $Q_i$  and  $|Q_{i-j}| = |Q_i| + \dots + |Q_j|, j > i$ . From  $V = Q$  above and Table 2, it follows that

$$2|Q_{13}| + |Q_5| + |Q_{10-12}| = Q \quad (2)$$

as the left-hand side counts the number of vertices in the quad mesh.

Again, as  $V = Q$  (approximately), the number of quads which have two free vertices must be equal to the number of quads which have no free vertices. Thus from Table 1, it follows that,

$$|Q_{13}| = |Q_{1-4}| + |Q_{6-9}| \quad (3)$$

Each branch in a quad spanning tree ends in a quad of type  $Q_1$  (a leaf node), and each branch begins either at the root gate or at a quad of type  $Q_3$  or

$Q_{7-11}$ . With each quad of type  $Q_3, Q_7, Q_9, Q_{10}$ , and  $Q_{11}$ , one more quad of type  $Q_1$  is associated. With each quad of type  $Q_8$ , two more quads of type  $Q_1$ 's are associated. Therefore, we have the following constraint.

$$|Q_3| + |Q_{7-11}| + |Q_8| = |Q_1| - 1 \quad (4)$$

Using the constraint of equation (3), we can pair a quad of each of the types  $Q_{1-4}$  and  $Q_{6-9}$  with a quad of the type  $Q_{13}$  as shown in Table 3 (Here and subsequently, we have taken the liberty of denoting a quad by its type).

Thus the grouping of quads of type  $Q_{1-9}$  with quads of type  $Q_{13}$  yield an average bit count of at most 3. Next, we use the constraint of equation (4) to refine this analysis even further. This constraint implies that  $|Q_{7-11}| + |Q_8| < |Q_1| - 1$ . Therefore, quads of each of the types from  $Q_{7-11}$  and  $Q_8$  can be associated with at most one quad of type  $Q_1$ .

Since quads of type  $Q_3$  have been taken care of, we don't have to find pairs for quads of this type. Since quads of type  $Q_1$  have been grouped with quads of type  $Q_{13}$  already, and a quad of type  $Q_1$  is associated with a quad of each one of the types  $Q_7, Q_9, Q_{10}$  and  $Q_{11}$ , while two quads of type  $Q_1$  are associated with a quad of type  $Q_8$ , we need to associate a quad group  $(Q_1, Q_{13})$  with each one of the quad groups/quads  $(Q_7, Q_{13}), (Q_9, Q_{13}), Q_{10}$ , and  $Q_{11}$ . Further, we need to associate two quad groups  $(Q_1, Q_{13})$  with one quad group  $(Q_8, Q_{13})$ . The grouping details are shown in Table 4:

The above grouping ensures that  $Q_7, Q_8, Q_9, Q_{10}, Q_{11}$  can be grouped to achieve an upper bound of at most 3 bits per vertex. Interaction-types  $Q_5$  and  $Q_{12}$  do not need to be grouped, since these are already assigned 3 bits each. We conclude that quadmesh connectivity can be encoded in less than 3 bits per vertex. Table 5 summarizes the final groupings.

From Table 3, the total cost of the encoding is

$$3Q - (Q_2| + |Q_3| + |Q_4|) - 3(|Q_7| + |Q_9|) - 6|Q_8| - |Q_{10}| - |Q_{11}| - 3|Q_3| - 3.$$

Since  $V = Q + 2$  (exactly) for a quad mesh, the total cost is therefore guaranteed to be less than 3 bits per vertex.

Encoding	Current Quad	Next Quad	Code	Num. of bits
Quad started with Q6-13	Q6	Q1-5	11111	5
		Q6-13	11110	5
	Q7	Q1-5	11101	5
		Q6-13	11100	5
	Q8	Q1-5	11011	5
		Q6-13	11010	5
	Q9	Q1-5	11001	5
		Q6-13	11000	5
	Q10	Q6-13	10111	5
	Q11	Q1-5	10110	5
		Q6-13	10101	5
	Q12	Q6-13	100	3
	Q13	Q6-13	0	1
Quad started with Q1-5	Q1	Q1-5	00	2
		Q6-13	01	2
	Q2	Q1-5	1100	4
		Q6-13	1101	4
	Q3	Q1-5	1010	4
		Q6-13	1011	4
	Q4	Q1-5	1000	4
		Q6-13	1001	4
	Q5	Q6-13	111	3

Table 2: *Coding Scheme*

## 4 Improving the upper bound to 2.67 bpv

Table 6 shows the connection between the labelling schemes of Kronrod-Gotsman[KG] and King et al[KRS]. The first row contains the quad labels, while the second row contains the equivalent combinations of CLRES labels. This correspondence is obtained by noticing that in the scheme of King et al [2] a quad is implicitly split by a diagonal into two triangles so that the next gate is situated counter-clockwise with respect to the current one.

From the above table of equivalence of labels, we can obtain an encoding scheme that uses less than 2.67 bits per vertex, using the encoding schemes of King et al [2].

## 5 Conclusions

In this note, we show that a scheme proposed by Kronrod & Gotsman [1] can be improved to have an upper bound of less than 3 bits per vertex. Also an easy equivalence between the labelling schemes of [1] and [2] shows that this can be further improved to 2.67 bits per vertex for manifold meshes. We believe that the upper bound can be further improved and this is the main open question.

## References

- [1] Boris Kronrod and Craig Gotsman. *Efficient coding of non-triangular mesh connectivity*, Proc. 16th European Workshop on Computational Geometry, pp. 24-26, 2000.

Code1	bits in Code1	Code2	bits in Code2	Average bits
$Q_1$	2	$Q_{13}$	1	1.5
$Q_2$	4	$Q_{13}$	1	2.5
$Q_3$	4	$Q_{13}$	1	2.5
$Q_4$	4	$Q_{13}$	1	2.5
$Q_6$	5	$Q_{13}$	1	3.0
$Q_7$	5	$Q_{13}$	1	3.0
$Q_8$	5	$Q_{13}$	1	3.0
$Q_9$	5	$Q_{13}$	1	3.0

Table 3: *Code bits analysis for  $Q_1$  to  $Q_9$*

Group	Total bits	Group	Total bits	Average bits
$(Q_7, Q_{13})$	6	$(Q_1, Q_{13})$	3	2.25
$(Q_8, Q_{13})$	6	$(Q_1, Q_{13}, Q_1, Q_{13})$	6	2.0
$(Q_9, Q_{13})$	6	$(Q_1, Q_{13})$	3	2.25
$Q_{10}$	5	$(Q_1, Q_{13})$	3	2.67
$Q_{11}$	5	$(Q_1, Q_{13})$	3	2.67

Table 4: *Code bits analysis for  $Q_7$  to  $Q_{11}$*

Grouping	Total Cost	quads in group	Amortized Cost	bits saved	occurrences of this group
$(Q_2, Q_{13})$ or $(Q_3, Q_{13})$ or $(Q_4, Q_{13})$	5	2	2.5	1	$ Q_2  +  Q_3  +  Q_4 $
$(Q_7, Q_{13}, Q_1, Q_{13})$ or $(Q_9, Q_{13}, Q_1, Q_{13})$	9	4	2.25	3	$ Q_7  +  Q_9 $
$(Q_8, Q_{13}, Q_1, Q_{13}, Q_1, Q_{13})$	12	6	2.0	6	$ Q_8 $
$(Q_{10}, Q_1, Q_{13})$	8	3	2.67	1	$ Q_{10} $
$(Q_{11}, Q_1, Q_{13})$	8	3	2.67	1	$ Q_{11} $
remaining( $Q_1, Q_{13}$ )	3	2	1.5	3	$ Q_3  + 1$
$Q_5$	3	1	3	0	$ Q_5 $
$(Q_6, Q_{13})$	6	2	3	0	$ Q_6 $
$Q_{12}$	3	1	3	0	$ Q_{12} $

Table 5: *Amortization Analysis*

KG	$Q_1$	$Q_2$	$Q_3$	$Q_4$	$Q_5$	$Q_6$	$Q_7$	$Q_8$	$Q_9$	$Q_{10}$	$Q_{11}$	$Q_{12}$	$Q_{13}$
KRS	$LE$	$LL$	$LS$	$LR$	$LC$	$LE$	$SL$	$SS$	$SR$	$SC$	$CS$	$CR$	$CC$

Table 6: *Correspondence between the labelling schemes*

- [2] Davis King, Jarek Rossignac and Andrzej Szmczak. *Connectivity compression for irregular quadrilateral meshes*, GVV Tech Report GVV-GIT-99-36.
- [3] M. Deering. *Geometry Compression*, Proc. ACM SIGGRAPH'95, pp. 13-20, August 1995.