

# **Geometric Compression**

**Leila De Floriani**

- Compression of geometric information
- Compression of connectivity: a taxonomy
- Triangle strips
- Techniques based on graph traversal
- Examples:
  - A simple compression technique (De Floriani et. al, 1998)
  - Edge-breaker (Rossignac, 1998)
- Progressive techniques
- Examples:
  - Progressive meshes (Hoppe, 1996)
  - Compressed progressive meshes (Pajarola and Rossignac, 2000)

## **Why Geometric Compression?**

- Availability of large geometric data sets in mechanical CAD, virtual reality, medical imaging, scientific visualization, geographic information systems, etc.
- Need for
  - speeding up transmission of geometric models
  - reducing the costs of memory and of auxiliary storage required by such models
  - enhancing rendering performances: limitations on on-board memory and on data transfer speed

## Why Geometric Compression?

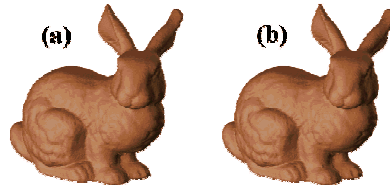
- Compression methods developed for triangle meshes (recent proposals for tetrahedral meshes)
- Compression methods aimed at two complementary tasks:
  - *compression of geometry*: efficient encoding of numerical information attached to the vertices (position, surface normal)
  - *compression of mesh connectivity*: efficient encoding of the mesh topology
- Geometric information play a very important role: they require more than 50% of the space needed to encode a mesh with  $2^{16}$  triangles
- In the case of connectivity information, it is necessary to store at least the vertices for each triangle (i.e., the Triangle-Vertex relation)

## Compression of vertex coordinates

- We consider a triangle mesh embedded in  $\mathbb{R}^3$ .
- Vertices are expressed as floating-point real numbers, which require 32 bits (24 bits for the mantissa) on common architectures, thus leading to 96 bits per vertex just to encode vertex coordinates (192 bits if we encode vertex coordinates plus normals).
- Compression of vertex information is performed by discretizing coordinates and expressing them on a predefined number of bits (lossy compression)
- Compression of vertex coordinates is performed in three stages (Deering, 1995):
  - *normalizing the mesh* in such a way to transform its axis-parallel bounding box into a cube with unit edge length (the 24 bits of the mantissa are thus the only meaningful information)

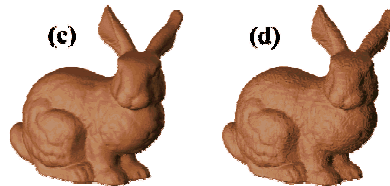
## Compression of vertex coordinates

- *discretizing coordinates*: coordinates are expressed as fixed point real numbers; they are discretized to  $q$  bits ( $q \leq 16$ ) by truncating the less significant  $m$  bits of position components, where  $m = 16 - q$



- *Example*:

- (a): original
- (b): 16 bits
- (c): 12 bits
- (d): 8 bits

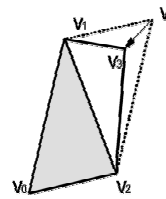
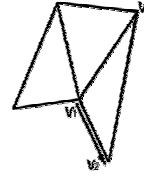


## Compression of vertex coordinates

- *predicting vertex positions*, i.e., just the difference between a vertex position and that of one or more its predecessor in the bit stream is encoded.
- Methods for predicting vertex positions:
  - Based on encoding the mesh as a triangle strip (Deering, 1995)
  - Parallelogram rule (Touma and Gotsman, 1998)
  - Tree-based prediction (Taubin and Rossignac, 1998)

## Compression of vertex coordinates

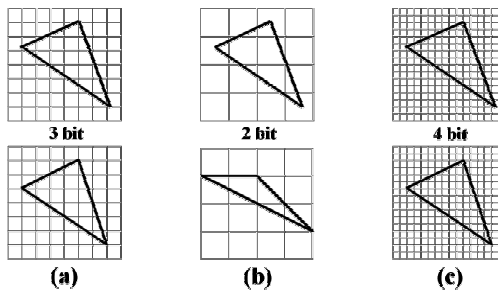
- *Deering's method:*
  - coordinates of vertex  $v_2$  are computed from those of vertex  $v_1$  and from vector  $v_1v_2$
  - length of the vector is several order of magnitude smaller than the width of the whole mesh (thus fewer bits are different from zero)
- *Parallelogram rule:*
  - Position of  $v_3$  is estimated as  $v'_3 = v_1 + v_2 - v_0$
  - Only vector  $v'_3 v_3$  must be encoded
  - Used in the MPEG-4



## Optimization: Chow's method

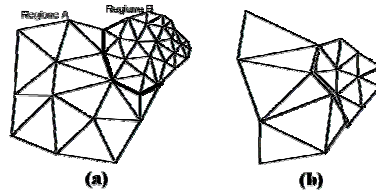
Number of bits used in the discretization stage: in some cases a larger number of bits increases the storage space required by the mesh, but it does not guarantee images of higher quality.

For instance, 3 bits are sufficient in the example below:



## Optimization: Chow's method

- Often triangle meshes have a non-uniform resolution
- Chow's method (1997) : subdivide the mesh into regions based on local levels of detail (computed as the average length of the mesh edges). Regions with a higher resolution are encoded with more bits



- Problems: inconsistencies among adjacent regions (Figure (b))

## Compressing connectivity

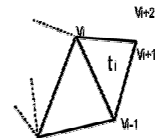
- We need techniques to encode topological relations in a compact way:
  - For visualization purposes, an implicit encoding of Triangle-Vertex relation is sufficient.
  - In general, we need an implicit encoding of both Triangle-Vertex and Triangle-Triangle relations
- Other important issues:
  - For transmission, we need some form of progressive encoding, as for images
  - Encoding triangle meshes with a non-manifold domain is necessary for CAD data

## Compressing connectivity

- Classification:
  - *Direct techniques*: minimize the number of bits needed to encode connectivity
  - *Progressive techniques*: an interrupted bit stream must provide a description of the whole object at a lower level of detail
- *Major direct techniques*:
  - Techniques based on triangle strips (for rendering application)
  - Techniques based on graph traversal

## Triangle strips

- Technique used by graphic libraries (GL, OpenGL)
- *Underlying idea*: if we sort the triangles of a mesh in such a way that two consecutive triangles share an edge, it is sufficient to give just the new vertex of a triangle instead of all its three vertices
- A sequence of triangles  $t_1, t_2, \dots, t_m$ , in which every pair of consecutive triangles  $t_i, t_{i+1}$  share an edge is called a (*generalized*) *triangle strip*
- It is not sufficient, in general, to specify a sequence of vertices to describe a generalized triangle strip: the next triangle in the figure can either be  $(v_i, v_{i+1}, v_{i+2})$  or  $(v_{i-1}, v_{i+1}, v_{i+2})$

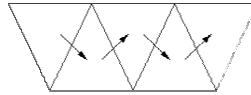


## Triangle strips

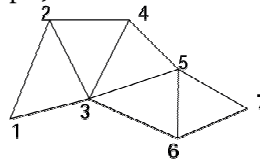
- Given a triangle mesh with  $m$  triangles, if it is possible to sort its triangles in such a way that the first triangle has vertices  $(v_0, v_1, v_2)$  and each pair of consecutive triangles  $t_i, t_{i+1}$  shares edge  $v_i v_{i+1}$ , then sequence  $t_1, t_2, \dots, t_m$  is called a *simple triangle strip*

- Examples:*

– A simple triangle strip



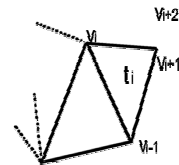
– A triangle simple (which is not simple)



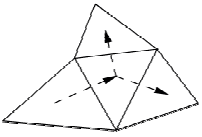
## Encoding a (generalized) triangle strip

*Two methods:*

- For each triangle  $t_{i+1}$ , the new vertex  $v_{i+2}$  is specified together with one bit (*swap code*). Depending on the value of the swap code, the pair  $(v_i, v_{i+1})$  or the pair  $(v_{i-1}, v_{i+1})$  is connected to  $v_{i+2}$  (GL library)
- The edge shared by triangles  $t_i$  and  $t_{i+1}$  is assumed to be  $v_i, v_{i+1}$ . In order to specify  $t_{i+1}$  as  $(v_{i-1}, v_{i+2}, v_{i+1})$ , we need to specify  $v_{i-1}$  again, creating the degenerate triangle  $(v_{i-1}, v_{i+2}, v_{i-1})$  (OpenGL library)
- Cost of the first method:* one vertex plus one bit per triangle (*one bit per vertex is the overhead due to the swap code*)



## Triangle strips: issues

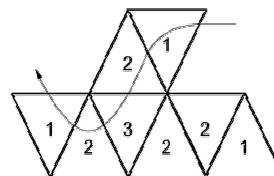
- *Result:* finding a triangle strip which covers a triangle mesh is equivalent to finding a Hamiltonian path in an undirected graph (i.e., a simple path which passes through all the nodes of a graph) in the dual graph of the triangle mesh, and, thus, it is NP-complete (Arkin et al., 1994)
- In this example no Hamiltonian path exists:
- *Result:* the problem of minimizing the number of strips is NP-hard (Evans et al., 1996)
- *Issues:*
  - define heuristics which tend to reduce the number of strips
  - reduce the length of the bit stream encoding the individual strips

Copyright 2002 Leila De Floriani

15

## Building triangle strips

- Well-known heuristics used to reduce the number of isolated triangles and to produce long strips (SGI algorithm).
- *SGI algorithm:*
  - For each triangle  $t$  of the mesh, compute the number of its adjacent triangles, that we call the *degree* of  $t$
  - We start with the triangle of minimum degree
  - At each step, the adjacent triangle to the current triangle  $t$  with the minimum degree is added. If this is not unique, we look ahead one step. If a decision cannot still be made, we choose randomly



Copyright 2002 Leila De Floriani

16



## Optimization and results

- *Deering's approach*: use a buffer of size 16 to store vertices already added to the bit stream. Instead of specifying a vertex, which has been already use, we can refer to its position in the buffer
- *Optimal buffer size* for a mesh with  $m$  triangles has been shown to be equal to  $12.27 \sqrt{m}$  (Ben-Yehuda and Gotsman, 1996) so that a vertex must be specified only once.
- *Compression results (experimental)*:
  - Deering's approach: 14.2 bits per triangle [28.4 bits per vertex]
  - Chow's approach: 10.1 bits per triangle [20.2 bits per vertex] (by using subdivision of the mesh into regions as well as a buffer)
- *Only Triangle-Vertex relation* is implicitly encoded in triangle strips
- All proposed techniques can encode manifold meshes with an arbitrary genus

## Techniques based on graph traversal

- They are based on the idea of encoding a traversal of a triangle mesh
- *Major benefits*:
  - High compression for connectivity information
  - Implicit encoding of adjacency information
- *Approaches*:
  - Topological surgery (Taubin and Rossignac, 1998)
  - Cut Border (Gumhold and Strasser, 1998)
  - Edge-breaker (Rossignac, 1998; and other papers from the same group at Georgia Tech.)
  - Method by Touma and Gotsman (1998)
  - Techniques for triangulated terrain surfaces (e.g., DeFloriani et al., 1998)

## A simple compression technique (De Floriani, Magillo, Puppò, 1998)

- It works for triangle meshes with a domain homeomorphic to a sphere or a disk
- Based on a *shelling order*: a sequence of all the triangles in the mesh with the property that the boundary of the set of triangles corresponding to any subsequence starting with the first triangle forms a simple polygon,
 

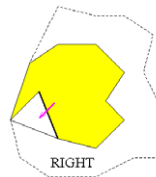
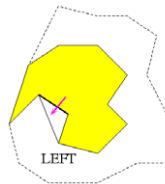
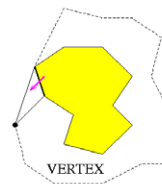
i.e., the sequence  $t_1, t_2, \dots, t_m$  of all the triangles in the mesh such that the union of the triangles in any subsequence  $t_1, t_2, \dots, t_q$ , where  $q \leq m$ , forms a polygonal region having a simple polygon as boundary
- The technique encodes the edges of the mesh. A code is associated with each edge: *Vertex, Left, Right, Skip*.

Copyright 2002 Leila De Floriani

19

## A simple compression technique

- *Vertex*: add a triangle with one edge on the current boundary polygon (add its other vertex as well)
- *Left/Right*: add a triangle which has two edges on the current boundary polygon

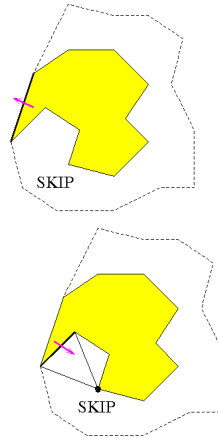


Copyright 2002 Leila De Floriani

20

## A simple compression technique

- *Skip*: the corresponding edge must be skipped since
  - it is on the boundary polygon
  - or
  - it would create a non-simple polygon (pseudo-manifold situation)

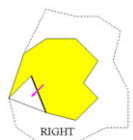
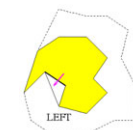
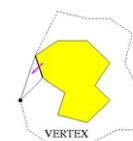


Copyright 2002 Leila De Floriani

21

## A simple compression technique: encoding algorithm

- *Approach*: grow a simple polygon  $P$  which bounds the triangles already processed. Keep a list of *active edges*, which are the edges of polygon  $P$  which have still to be encoded.
- Start from an arbitrary triangle. The three edges of the triangle form the boundary polygon  $P$ .
- *Loop* on the edges of the  $P$ : for each active edge  $e$ :
  - if* the triangle  $t$  externally adjacent to  $e$ , i.e., sharing edge  $e$  with  $t$  and not internal to  $P$  exists and, if  $t$  has only one edge on  $P$  and its other vertex is not on  $P$ , *then*
    - send a VERTEX, LEFT or RIGHT code as appropriate (see next slides)



Copyright 2002 Leila De Floriani

22

## A simple compression technique: encoding algorithm

- update the current polygon P by replacing the edge(s) of  $t$  on P with the edge(s) of  $t$  not on P
  - update the active edge list by replacing the edge(s) of  $t$  on P which are active with the edge(s) of  $t$  not on P
- else send a SKIP code; delete edge  $e$  from the active list

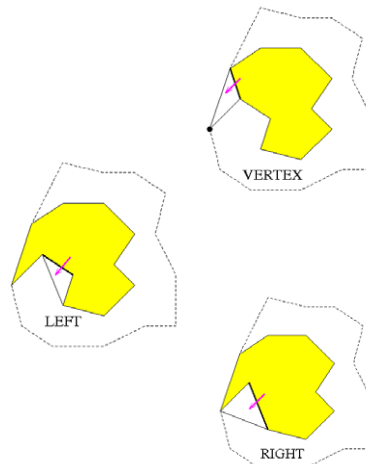


Copyright 2002 Leila De Floriani

23

## A simple compression technique: encoding algorithm

- if  $t$  brings a new vertex  
==> *Vertex* + vertex coordinates
- if  $t$  shares the edge of P preceding  $e$  in cw order ==> *Left*
- if  $t$  shares the edge of P following  $e$  in cw order ==> *Right*



Copyright 2002 Leila De Floriani

24

## **A simple compression technique: properties**

- Every vertex is encoded only once
- The encoding algorithm works in time linear in the size of the mesh, since (it can be proven that) an edge is at most examined once
- The decoding algorithm just plays back the encoding one
- Adjacencies between triangles (Triangle-Triangle relation) are reconstructed directly from the sequence at no additional cost
- *Cost of the codes:*
  - a 2-bits code per edge
  - *theoretically:* an upper bound of  $6n$  bits for encoding the connectivity a mesh with  $n$  vertices; at most 6 bits per vertex
  - *in practice:* between 4.1 and 4.4 bits per vertex (without any further compression on the bit stream), since some edge is not encoded

## **Edge-breaker (Rossignac, 1998)**

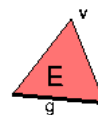
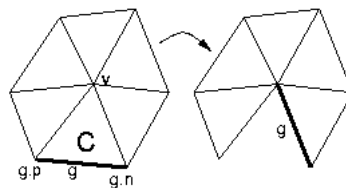
- It works for meshes with a manifold domain.
- We describe it for meshes with a domain homeomorphic to a sphere or a disk.
- A code is associated with the triangles of the mesh
- Encoding and decoding algorithms are based on a traversal of the mesh
- Encoding the mesh requires two phases:
  - traverse the mesh and generate a sequence of symbols which describe the operations performed
  - generate a binary encoding of the symbolic sequence based on the frequency of symbols in the sequence

## Edge-breaker: encoding algorithm

- Mesh traversal starts from the initial triangle mesh and eliminates one triangle at a time
- Several independent connected regions can be generated as a consequence of a triangle removal. Each region  $R_i$  is bounded by a simple polygon  $B_i$
- For each polygon  $B_i$ , a reference edge, denoted  $g_i$ , is considered. All reference edges are stored in a stack  $K$ . The edge  $g$  on top of the stack is the *active* edge
- Generic step of the algorithm:
  - Let  $R$  be the region containing  $g$  and let  $B$  denote the boundary of  $R$ . Let  $t$  be the triangle of  $R$  bounded by  $g$ . Let  $v$  be the vertex opposite to  $g$  on  $t$ .
  - Based upon the relation among,  $v$ ,  $g$  and  $B$ , five cases arise:

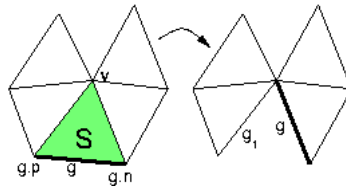
## Edge-breaker: encoding algorithm

- $v \notin B$ :  $v$  is inside region  $R$  and must be encoded.
  - send code  $C$  (*Code*)
  - replace edge  $g$  in  $B$  with edges  $g.pv$  and  $vg.n$
  - edge  $vg.n$  becomes the new active edge  $g$
- $v \in B$  and  $vg$  precedes and  $gv$  follows  $g$ : last triangle of region  $R$ 
  - send code  $E$  (*End*)
  - If  $K$  is empty, then the algorithm terminates, otherwise the next region is considered



## Edge-breaker: encoding algorithm

- $v \in B$  and  $v$  is not adjacent to  $g$ :
  - send code  $S$  (*Split*)
  - subdivide the current region and continue on each region independently
  - edge  $g_1 = g.pv$  is inserted in  $K$
  - edge  $vg.n$  becomes the new active edge  $g$

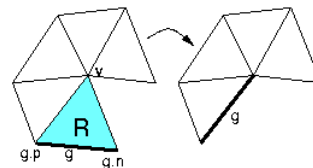
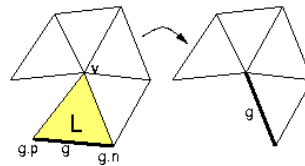


Copyright 2002 Leila De Floriani

29

## Edge-breaker: encoding algorithm

- $v \in B$  and  $v$  is adjacent to  $g.p$ :
  - send code  $L$  (*Left*)
  - delete edges  $vg.p$  and  $g$  from  $K$
  - insert edge  $g.nv$  on  $K$
  - edge  $g.nv$  becomes the new active edge  $g$
- $v \in B$  and  $v$  is adjacent to  $g.n$ :
  - send code  $R$  (*Right*)
  - delete edges  $g.nv$  and  $g$  from  $K$
  - insert edge  $vg.p$  on  $K$
  - edge  $vg.p$  becomes the new active edge  $g$



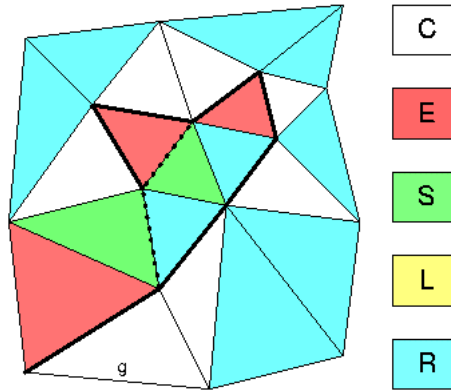
Copyright 2002 Leila De Floriani

30

## Edge-breaker: an example

Sequence:

CCRRRCRCRR  
CCRRCSERS  
REE



Copyright 2002 Leila De Floriani

31

## Edge-breaker

- The length of the sequence of symbols is equal to the number of triangles ( $= 2n$  for a mesh with a domain homeomorphic to the sphere or to the disk)
- *Code*: the five symbols can be encoded with 3 bits
- The length of the bit stream can be  $6n$  in the worst case (for meshes as defined above)
- To produce a compact bit stream, an expansion encoding is generated by evaluating the frequency of the different symbols
- For instance, if the number of vertices on the mesh boundary is small, we can use a compact encoding in which C=0, L=100, R=101, S=110, E=111. This leads to 4 bits per vertex since C (the most common case) requires just one bit.

Copyright 2002 Leila De Floriani

32



## Edge-breaker: discussion

- The five symbols are sufficient only for meshes with null genus. Rossignac has proposed an extension for dealing with meshes with a manifold domain by adding two other symbols.
- Adjacencies between triangles (Triangle-Triangle relation) can be reconstructed directly from the sequence.
- *Experimental results* have shown costs between 3.2 and 3.67 bits per vertex.
- The encoding scheme proposed by King and Rossignac (1999) *guarantees* a cost lower than 3.67 bits per vertex.

## Progressive techniques

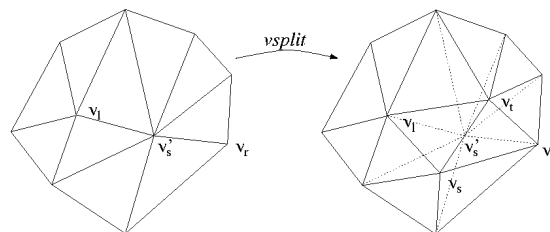
- They are aimed at progressive transmission of a mesh:
  - a coarse mesh approximation must be provided in a short time
  - the initial approximation is refined through subsequent incremental modifications
- They can be classified based on signal processing (wavelet methods) and geometric techniques
- Geometric techniques classified based on the granularity of the modifications

## Progressive techniques

- *Fine-grained progressive techniques*: based on modifications that update few triangles at the time; they produce more levels of detail
  - Progressive meshes (Hoppe, 1996)
- *Coarse-grained progressive techniques*: based on batches of modifications which are encoded at the same time; they usually lead to more compact bit streams
  - Progressive Forest Splits (Taubin et al., 1998)
  - A progressive technique based on vertex insertion (Cohen-Or et al., 1998)
  - Compressed progressive meshes (Pajarola and Rossignac, 2000)

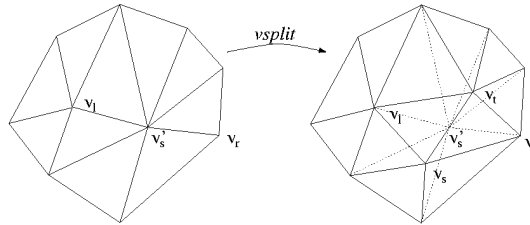
## Progressive Meshes (PMs) (Hoppe, 1996)

- A triangle mesh is encoded as a coarse mesh  $\Sigma_0$  plus a sequence of vertex splits (inverse of edge collapses)
- It is a fine-grained technique since at each refinement step at most two triangles are inserted



## Progressive Meshes

- Performing a vertex split requires:
  - update the coordinates of  $v'_s$  to get  $v_s$
  - insert the new vertex  $v_t$
  - add the two triangles  $v_s, v_l, v_t$  and  $v_s, v_t, v_r$



Note that the decoding will generate an indexed structure

## Progressive Meshes

- Specification of the four vertices would require four indexes on  $4 \log n$  bits, if  $n$  is the number of vertices of the mesh at full resolution
- *Improvements:*
  - It is possible to make a permutation of the vertex indexes in such a way that the  $n_0$  vertices of  $\Sigma_0$  have indexes between 0 and  $n_0 - 1$ , and  $t_i$  (the index of the vertex introduced by the  $i$ -th split) is equal to  $n_0 + i$ . Thus,  $t_i$  is not required.
  - Since  $v_l$  and  $v_r$  are vertices adjacent to  $v_s$ , it is sufficient to identify them among the vertices adjacent to  $v_s$ . Thus,  $2 \log 6 = 6$  bits are sufficient, since on average a vertex had six adjacent vertices
- Encoding a vertex split thus requires  $\log n + 6$  bits.

## Progressive Meshes

- The cost of encoding connectivity is equal to  $(\log n + 6)n$  bits, disregarding the space required for encoding  $\Sigma_o$ . If  $n=2^{20}$ , 26 bits per vertex.
- Techniques for encoding geometric information are different from those adopted by non-progressive methods: we need to add vertex  $v_t$  and move vertex  $v'_s$
- When dealing with an edge collapse to the mid-point, it is sufficient to encode vector  $v'_s v_t$ :  $v_s$  can be obtained as  $2v'_s - v_t$
- Experiments from Hoppe show that the space required for geometric information is between  $30n$  and  $37n$  bits