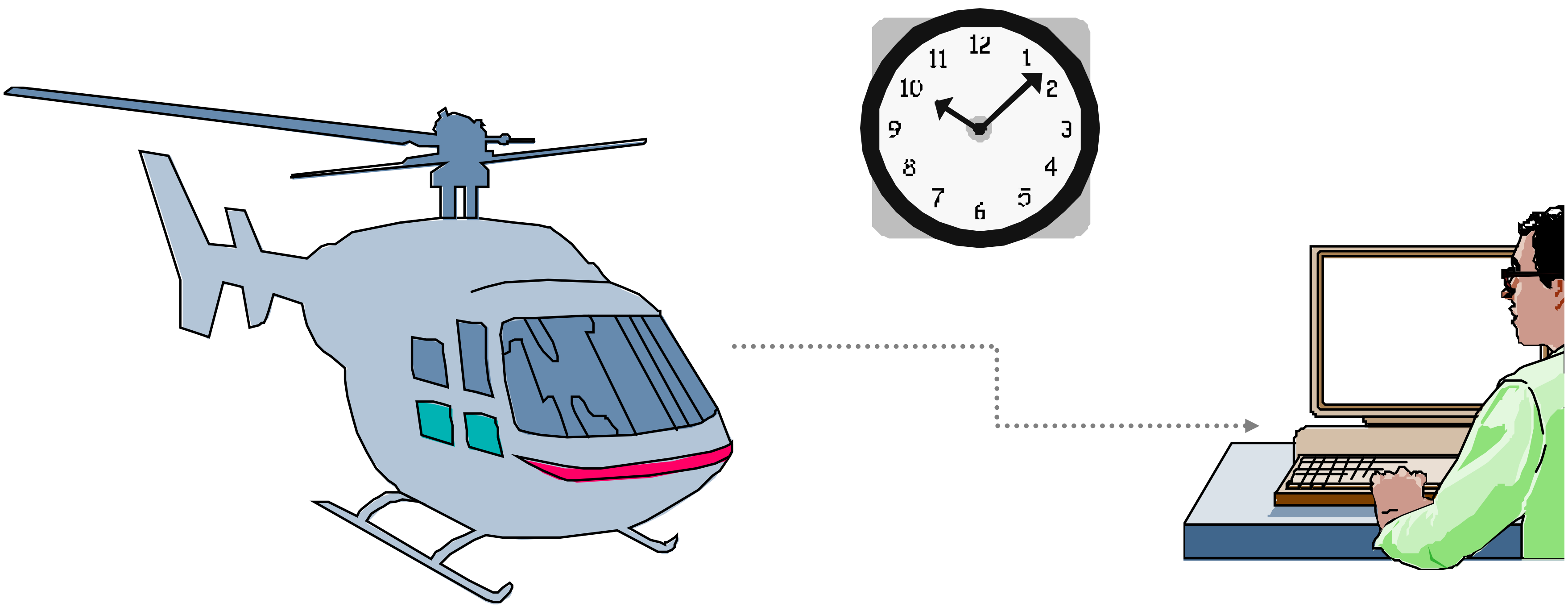
Mesh Compression

Ligang Liu
Graphics&Geometric Computing Lab
USTC
http://staff.ustc.edu.cn/~lgliu

Problem

- Delays in accessing 3D mesh models
  - Online games, viewer, search engine…

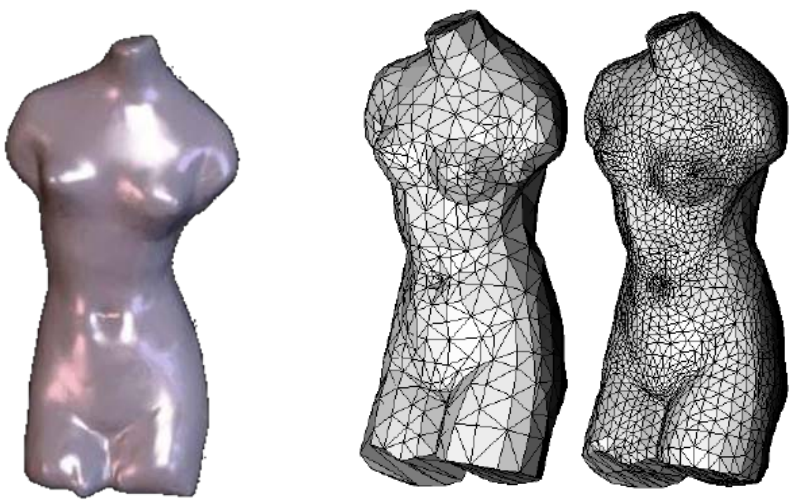


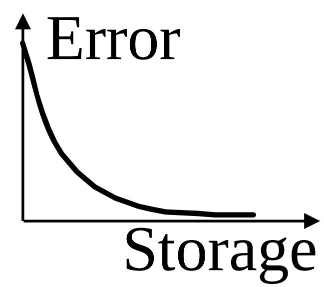
Motivation

- Bandwidth
  - Communicate large complex & highly detailed 3D models through low-bandwidth connection (e.g. VRML over the Internet)
- Storage
  - Store large & complex 3D models (e.g. 3D scanner output)

Storage size depends on

- The shape, topology, and attributes of the model
- Choice of representation
- Acceptable accuracy loss
- Compression used

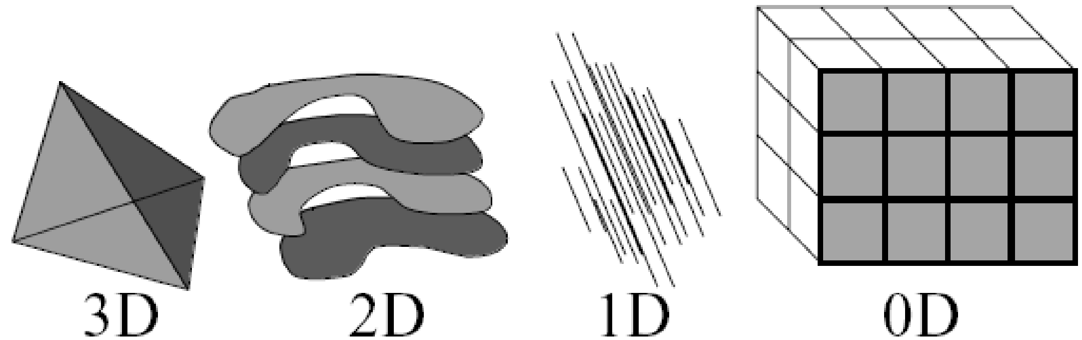




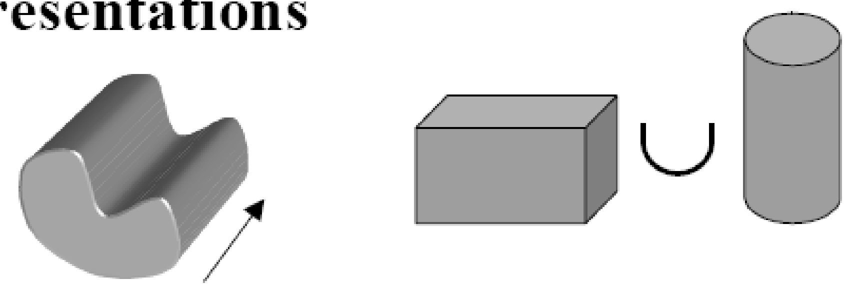
Representations of 3D Models

- **Volume decomposition**
  - <u>**Tetrahedra**</u>
  - Extrusions (slices, rays)
  - Voxels (octrees)

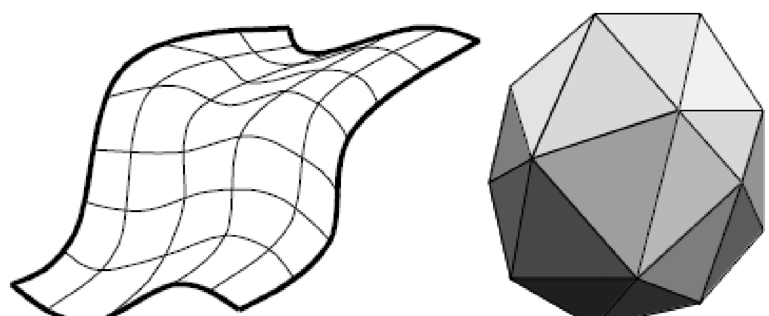
(Regularly) spaced samples: 3D primitives



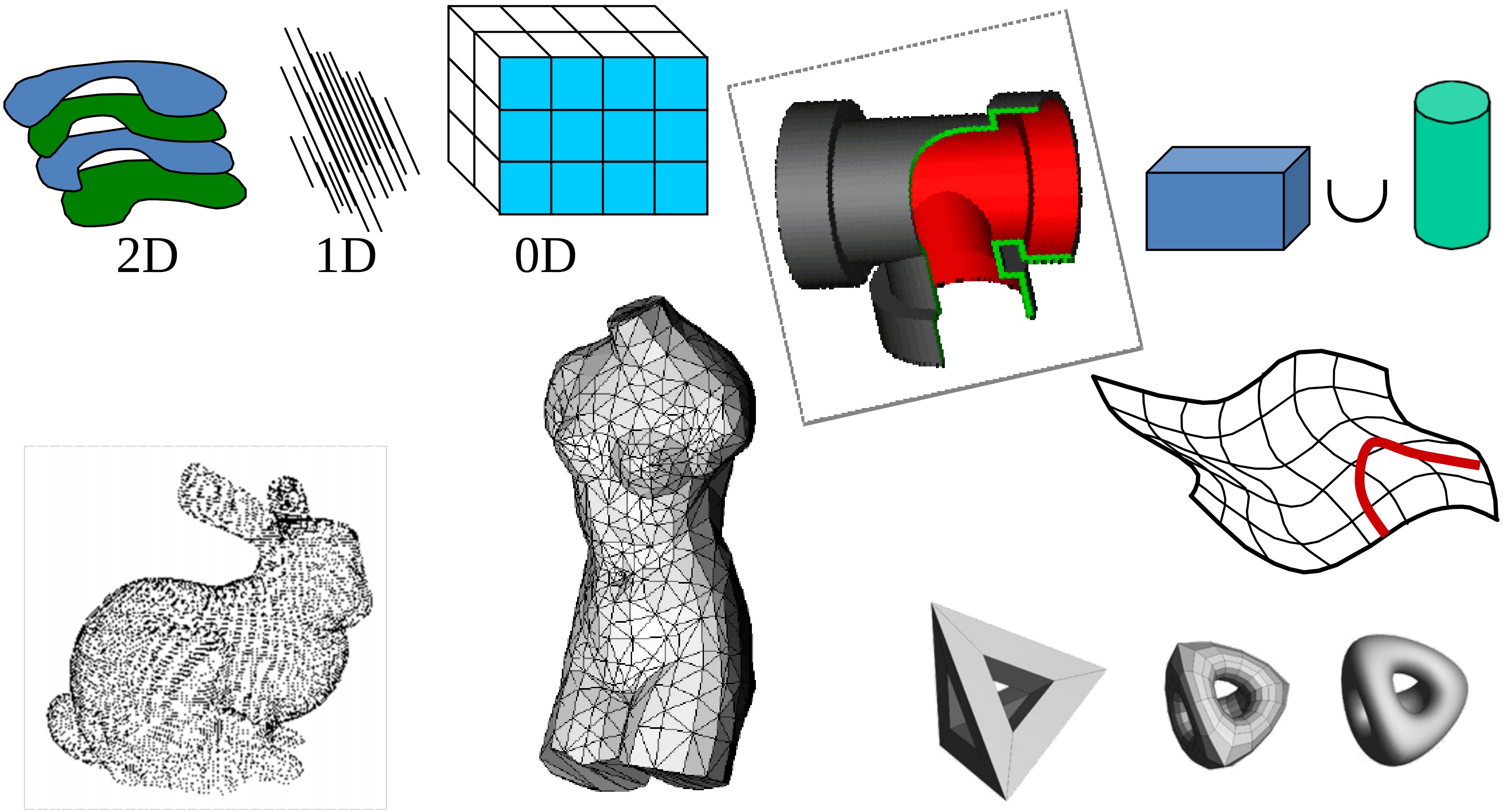
- **Procedural (constructive) representations**
  - CSG, R-sets
  - Sweeps, Minkowski sums



- **Boundary decomposition**
  - Parametric patches
  - <u>**Triangles**</u>, polygons, quads

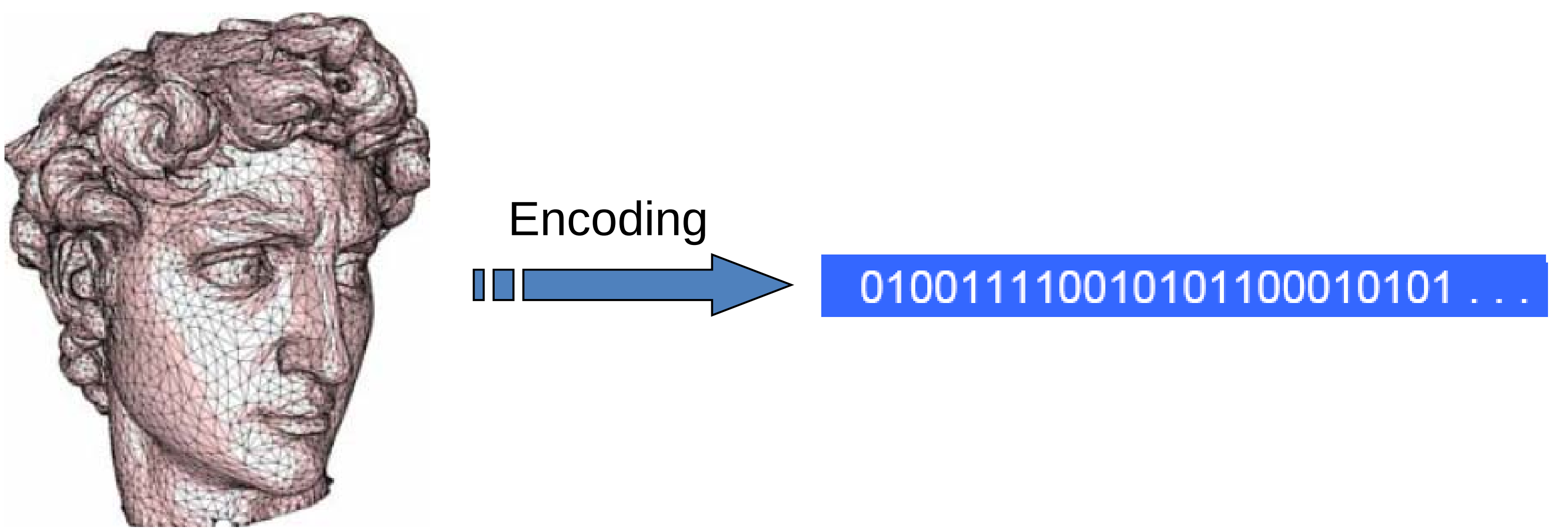


Storage size depends on representation



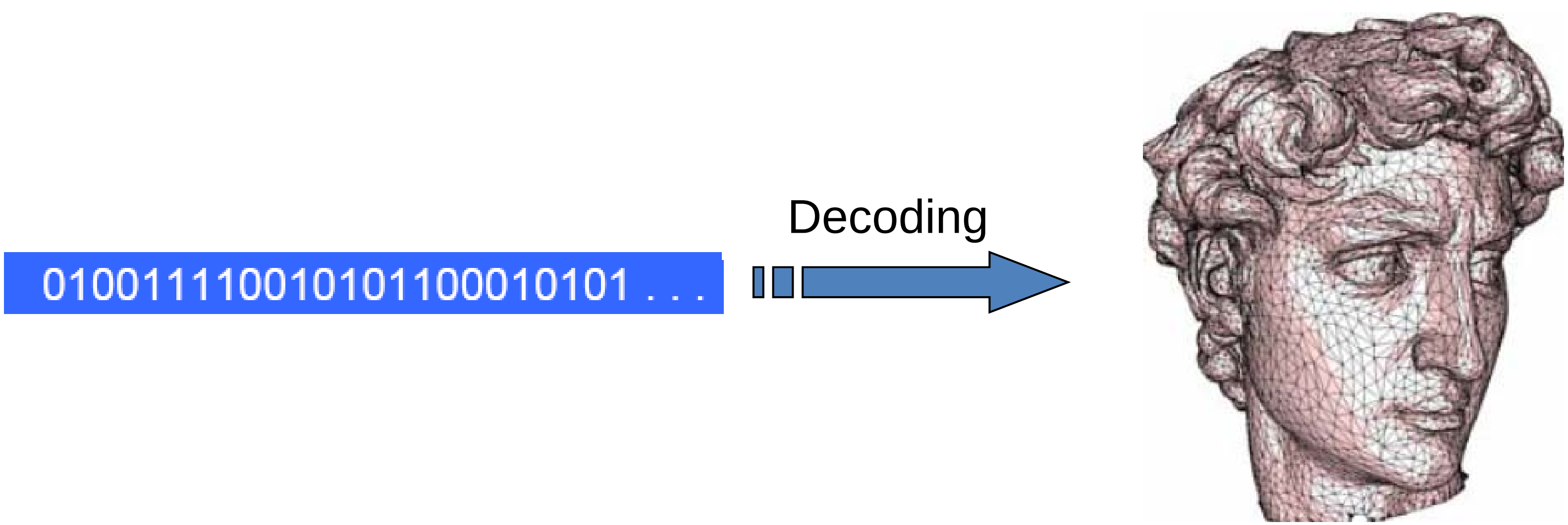
Mesh Encoding

- Input: 3D triangular mesh
  - Assumed to be orientable manifold
- Output: bit stream
  - 0100111100101011000101 . . .



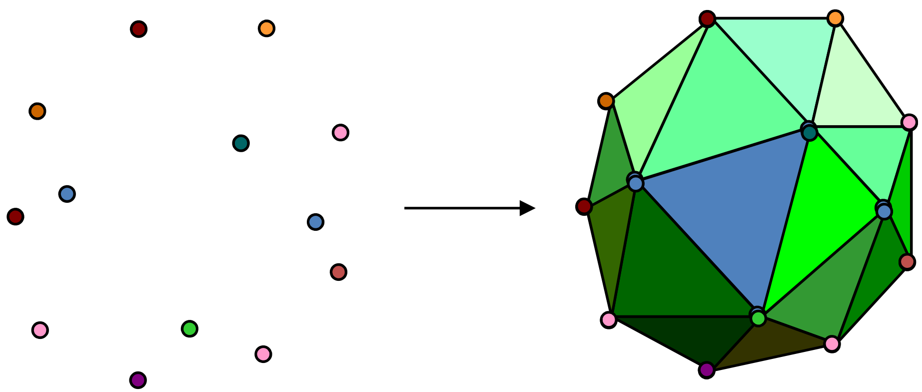
Mesh Decoding

- Input
  - Bit stream
- Output
  - Reconstruction of original 3D triangular mesh

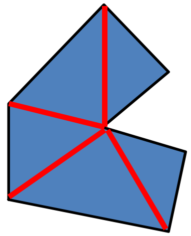


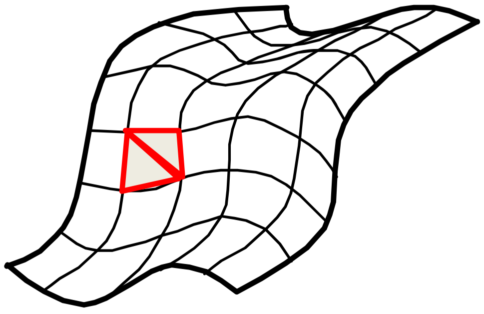
Why triangles and tetrahedra?

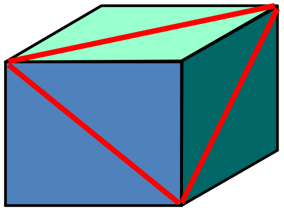
Triangles and **tetrahedra** are the simplest ways of specifying how **irregular point-samples** and associated values (color, density...) should be **interpolated** to approximate (non-homogeneous) sets.



Other representations may be easily triangulated/tetrahedralized.







Representing Triangle and Tetrahedra Meshes

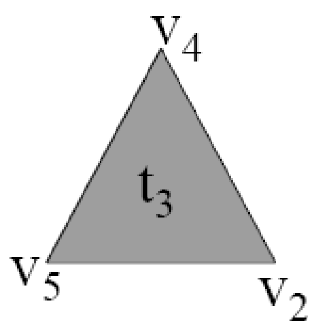
Vertices and values:
3x16+k bits/vertex

vertex 1	x	y	z	c
vertex 2	x	y	z	c
vertex 3	x	y	z	c

Triangle/vertex incidence:
$3x\log_2(V)$ bits/triangle

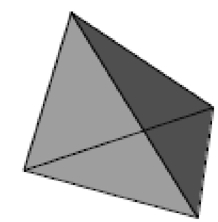
Triangle 1	1	2	3
Triangle 2	3	2	4
Triangle 3	4	2	5
Triangle 4	7	5	6
Triangle 5	6	5	8
Triangle 6	8	5	1

$T = 2V$



Tetrahedron/vertex incidence:
$4x\log_2(V)$ bits/tetrahedron

Tetrahedron 1	1	2	3	4
Tetrahedron 2	3	2	4	6
Tetrahedron 3	4	2	5	8
Tetrahedron 4	7	5	6	2
Tetrahedron 5	6	5	8	4
Tetrahedron 6	8	5	1	5
Tetrahedron 7	1	2	3	6
Tetrahedron 8	3	2	4	5
Tetrahedron 9	4	2	5	2
....				
Tetrahedron 17	6	5	8	1
Tetrahedron 18	8	5	1	2



$T \sim 6.5V$

Connectivity dominates storage cost!

Bandwidth Requirements for Triangular Meshes

- Naïve representation of a triangle mesh
  - Each triangle is represented by 3 vertices
    - Each vertex is represented by 3 coordinates
      - Each coordinate is represented by a float
- Total storage = 576 bits per vertex (bpv) for geoemtry
  - 3x3x32 bits per triangle
  - Twice as many triangles as vertices
- Not counting colors, normals, textures, motions

Two Parts for 3D Meshes

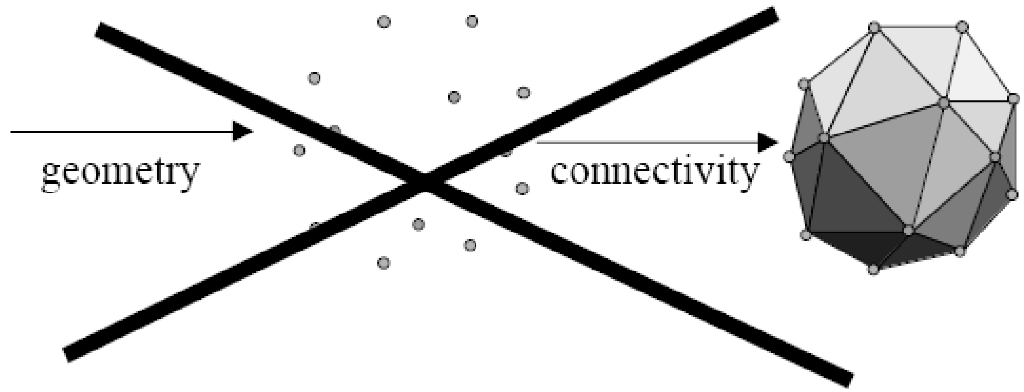
- Geometry
  - Coordinates of the vertices
  - v x y z
- Connectivity
  - How the vertices connect with each other
  - f i j k
- T = 2V

Mesh Compression

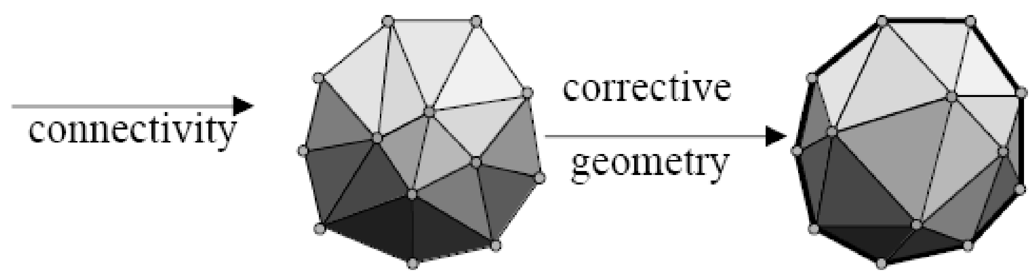
- Geometry encoding
- Connectivity encoding
- Which one should be done first?

Must Decode Connectivity First

Cannot use geometry to estimate connectivity, because connectivity is used to **predict** geometry (see vertex-data compression section).



Must develop **connectivity compression** methods **independent of the vertex locations**.



Two Categories

- Single resolution
  - Edge breaker, Topological surgery…
  - not transmission friendly.
- Multi resolution
  - CPM, PFS, VD…
  - transmission friendly

Discussions

Compression Theory

Coding Techniques

Coding Techniques

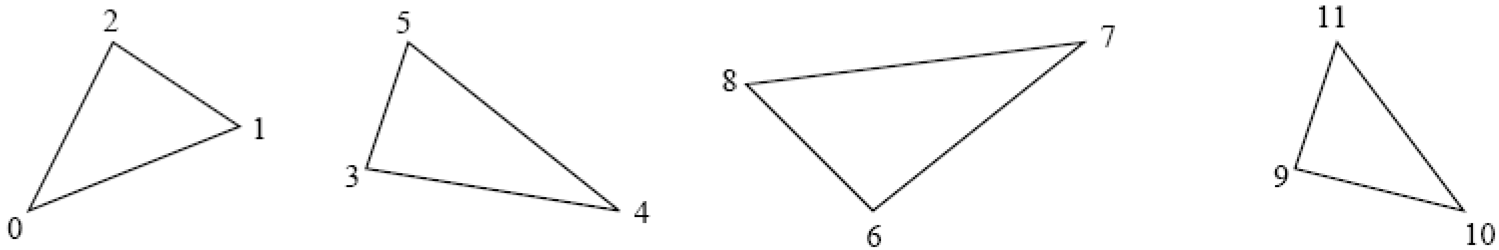
- RLE: Run Length Encoding
- LZW coding
- Huffman coding
- Arithmetic coding

Connectivity Encoding

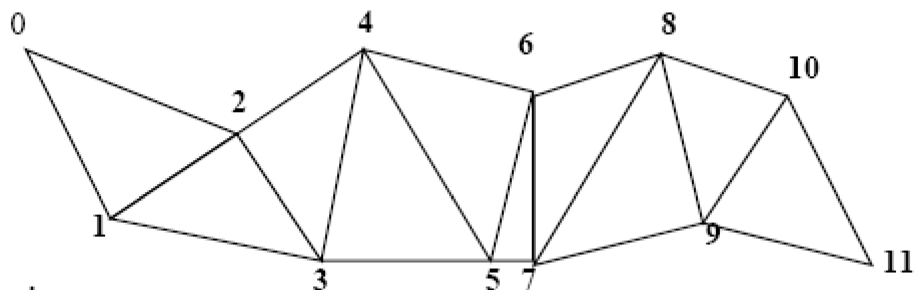
Connectivity Compression: An Old Problem

- **Use vertex permutation to encode incidence**
  - Denny,Sohler: **Encoding a triangulation as a permutation of its point set,** CCCG, 97
- **Compression of the connectivity graph (planar triangle graph)**
  - Itai,Rodeh: **Representation of graphs,** Acta Informatica, 82
  - Turan: **On the succinct representation of graphs,** Discrete Applied Math, 84
  - Naor: **Succinct representation of general unlabeled graphs,** Discrete Applied Math, 90
  - Keeler,Westbrook: **Short encoding of planar graphs and maps,** Discrete Applied Math, 93
  - Deering: **Geometry Compression,** Siggraph, 95
  - Taubin,Rossignac: **Geometric compression through topological surgery,** ACM ToG, 98
  - Taubin,Horn,Lazarus,Rossignac: **Geometry coding and VRML,** Proc. IEEE, 98
  - Touma,Gotsman: **Triangle Mesh Compression**, GI, 98
  - Gumbold,Straßer: **Realtime Compression of Triangle Mesh Connectivity,** Siggraph, 98
  - Rossignac: **Edgebreaker: Compressing the incidence graph of triangle meshes,** TVCG, 99
  - Rossignac,Szymczak: **Wrap&Zip: Linear decompression of triangle meshes,** CGTA, 99
  - Szymczak,Rossignac: **Grow&Fold: Compression of tetrahedral meshes,** ACM SM, 99
- **Compressed inverse of progressive simplification steps or batches**
  - Hoppe: **Progressive meshes,** Siggraph, 96
  - Taubin,Gueziec,Horn,Lazarus: **Progressive forest split compression,** Siggraph, 98
  - Pajarola,Rossignac: **Compressed Progressive Meshes,** IEEE TVCG99
  - Pajarola,Szymczak,Rossignac: **ImplantSpray: Compressed Tetrahedral Meshes,** VIS 99

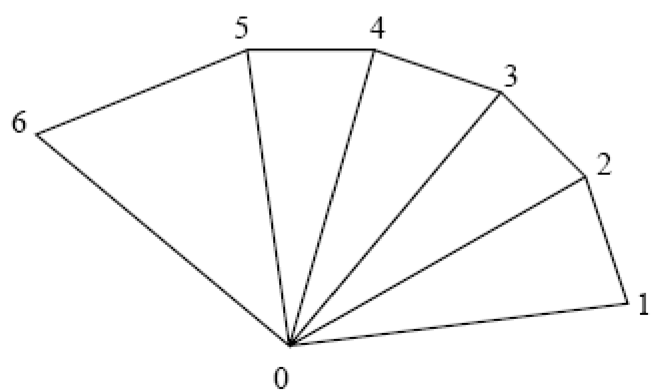
Simple Triangle Strips



Isolated Triangle Strip



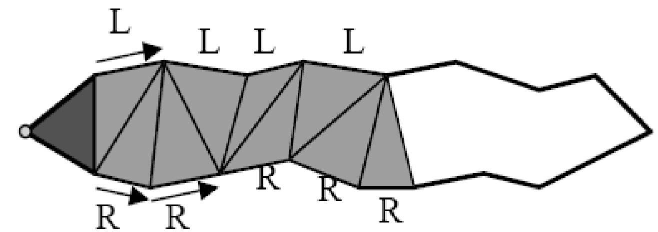
Zig-zag Triangle Strip



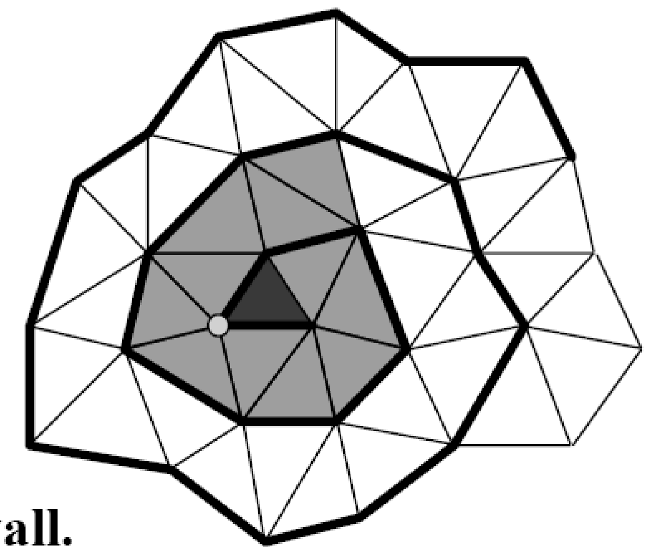
Star Triangle Strip

Case 1:
Spiraling Walls and Corridors

Given the left and right boundaries of a triangle strip (**corridor**), we need T (left/right) bits to encode its triangulation. ex: LRRLLRLRR

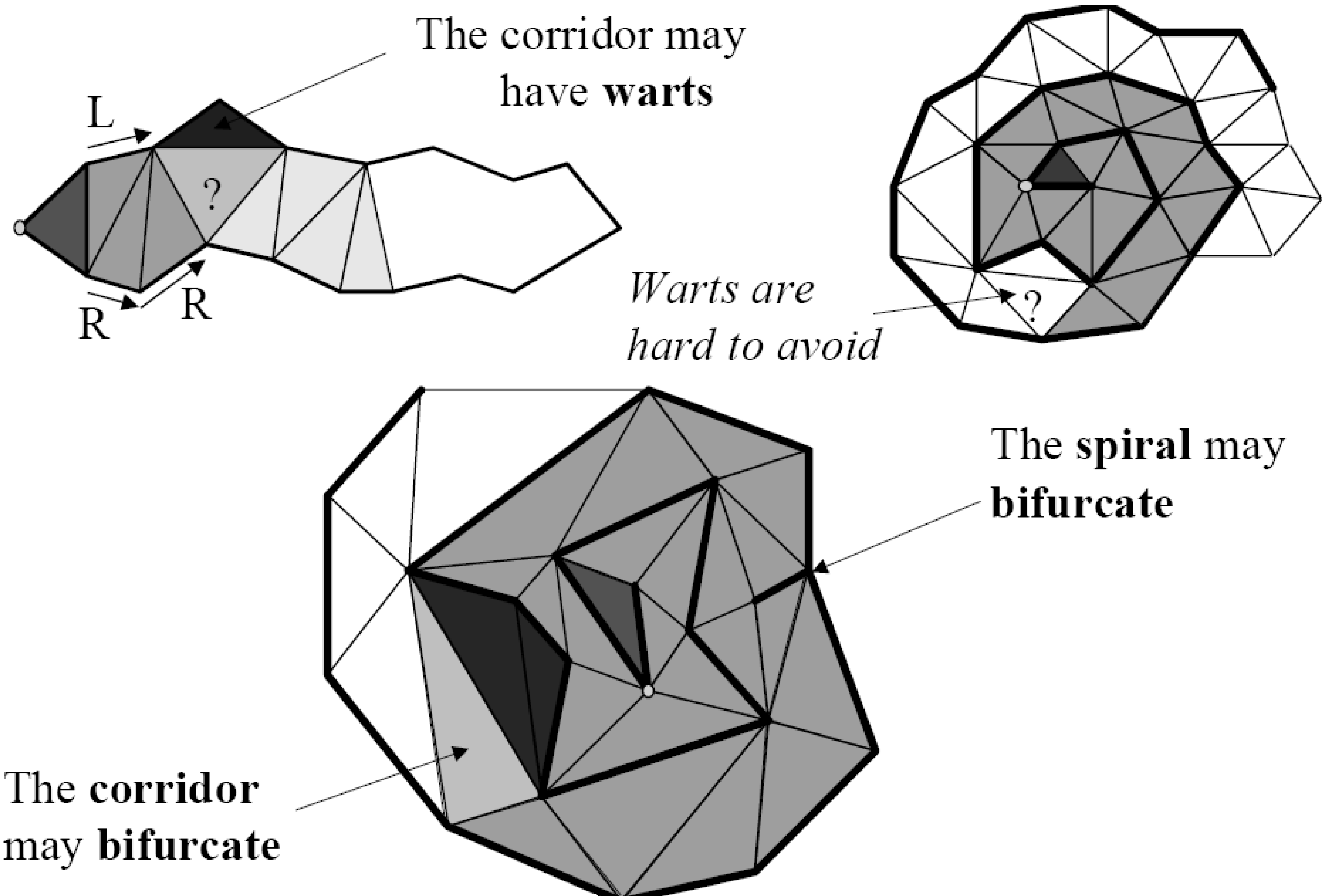


Connecting vertices into a single **spiral** (Hamiltonian walk) defines the left and the right boundaries (**walls**) of a long corridor.



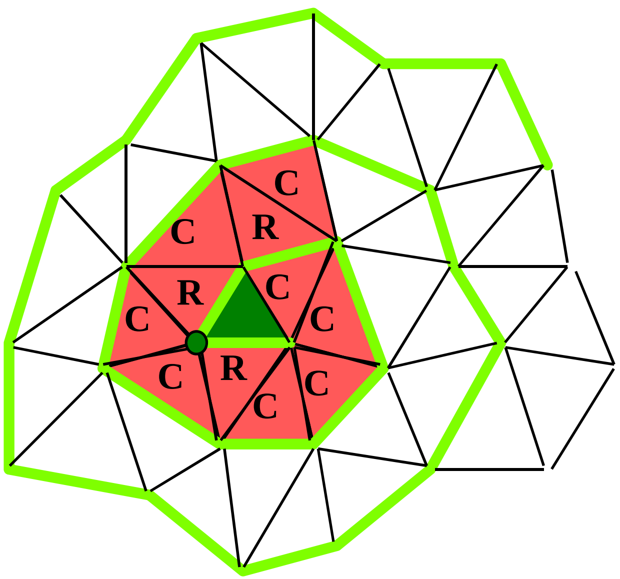
Store **vertices** in their **order along** the **wall.**
(Can use former vertices to predict location of new ones.)
Encode **connectivity** using only **1** (left/right) **bit per triangle** !

Problem

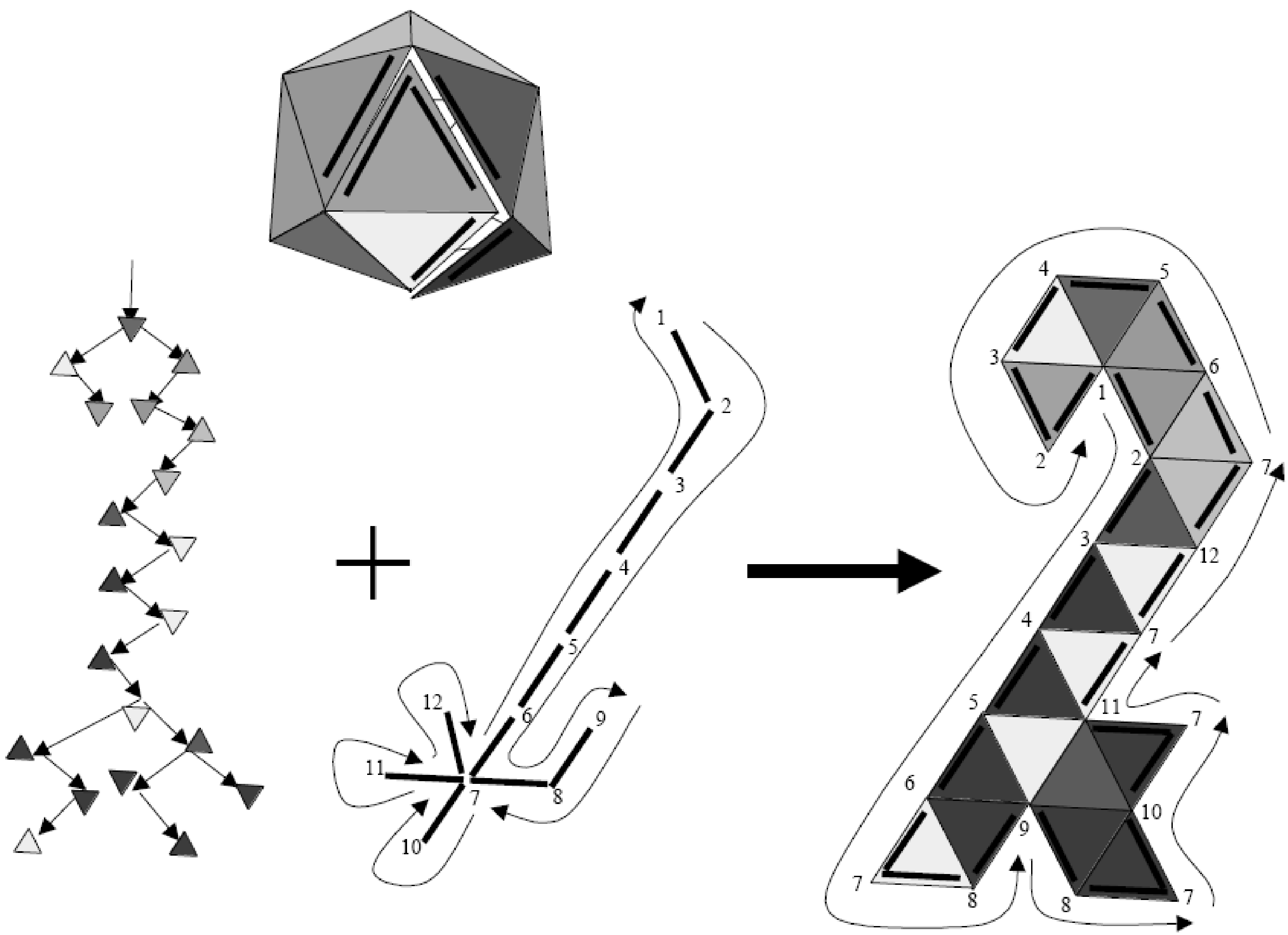


The corridor may have **warts**
L
R
R
?
Warts are hard to avoid
?
The **spiral** may **bifurcate**
The **corridor** may **bifurcate**

Example



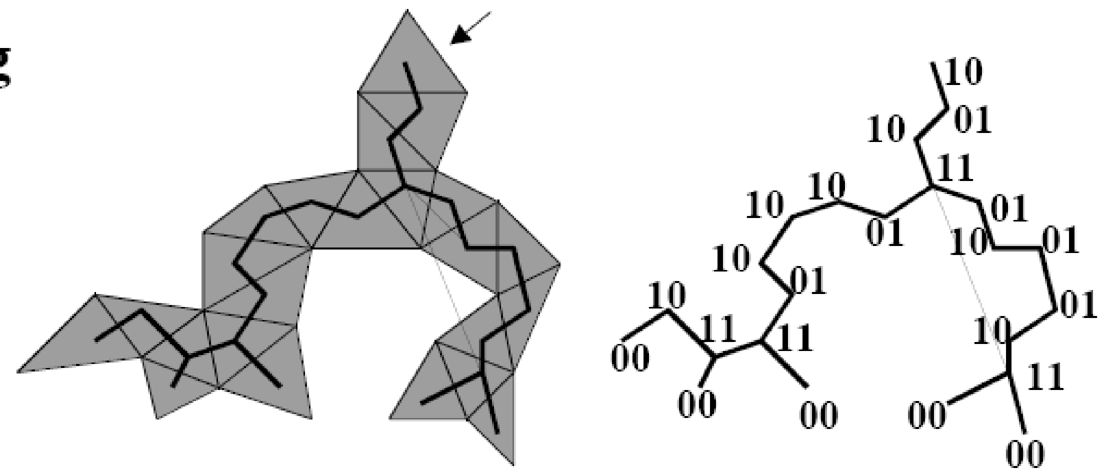
Case 2:
Triangle-tree & vertex-tree

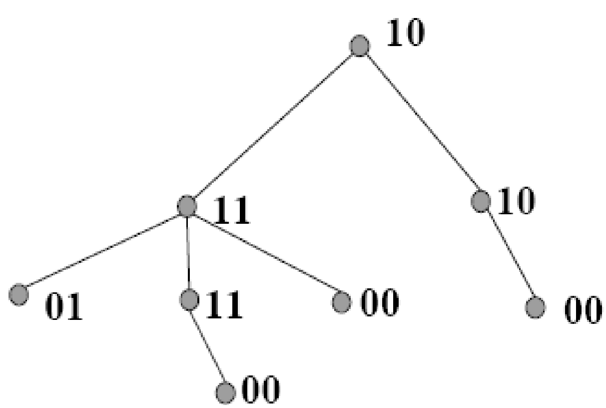


3T bit encoding

The triangle spanning tree may be encoded using 2T bits:

- **has-left-child**
- **has-right-child**

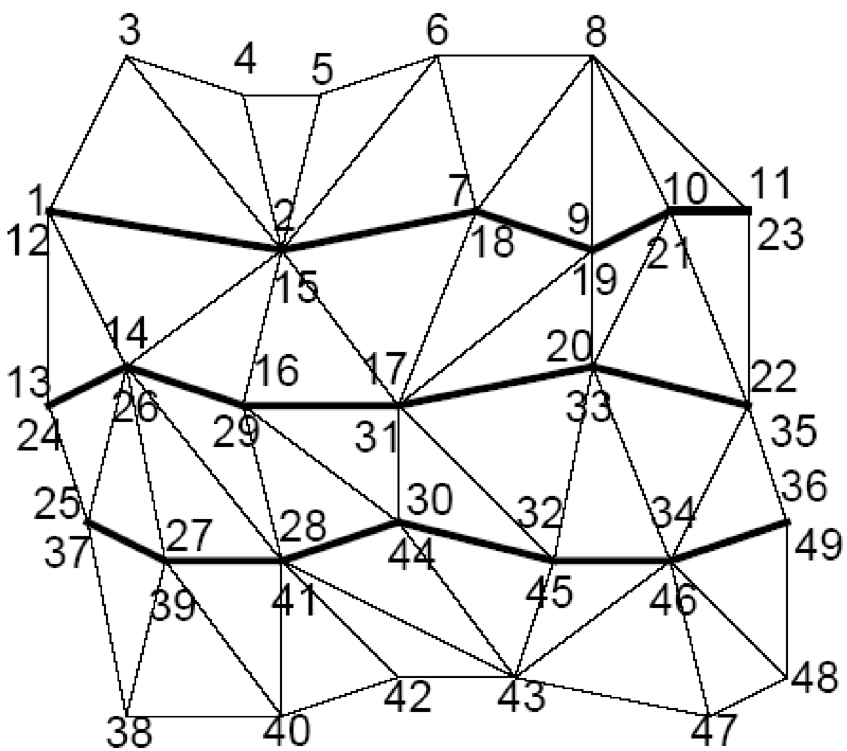


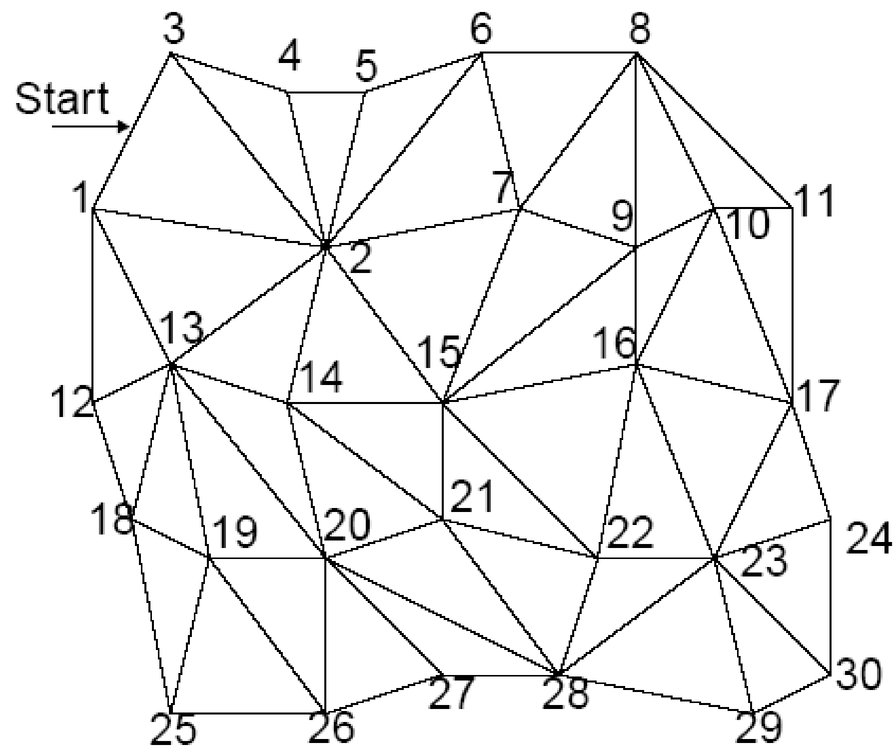


A vertex spanning tree may be encoded using 2V bits (= 1T bits):

- **has-children**
- **has-right-sibling**

Generalized Triangle Strip & Generalized Triangle Mesh





Case Study: Valence-based Codes

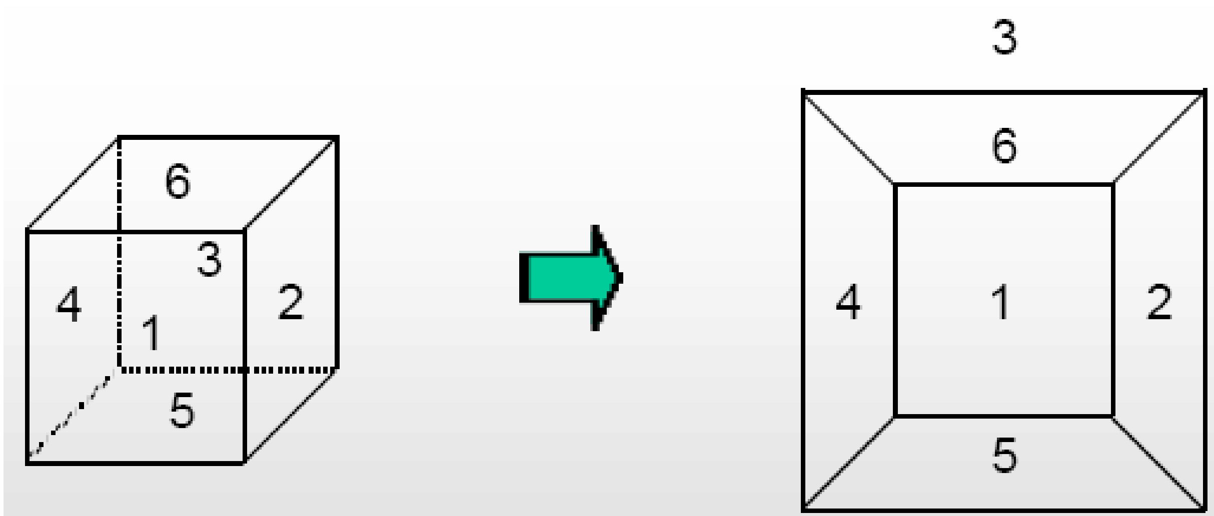
Touma C. and Gotsman C.

Triangle Mesh Compression.

Graphics Interface 1998

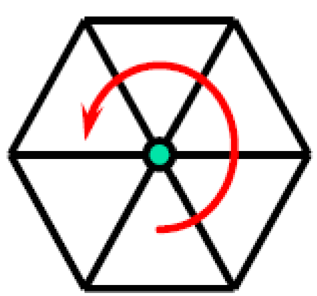
Key Principle

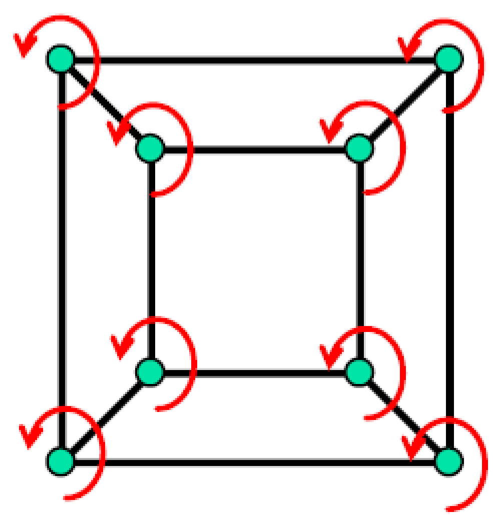
- A genus-0 manifold mesh is topologically equivalent to a planar graph.



Key Principle (cont.)

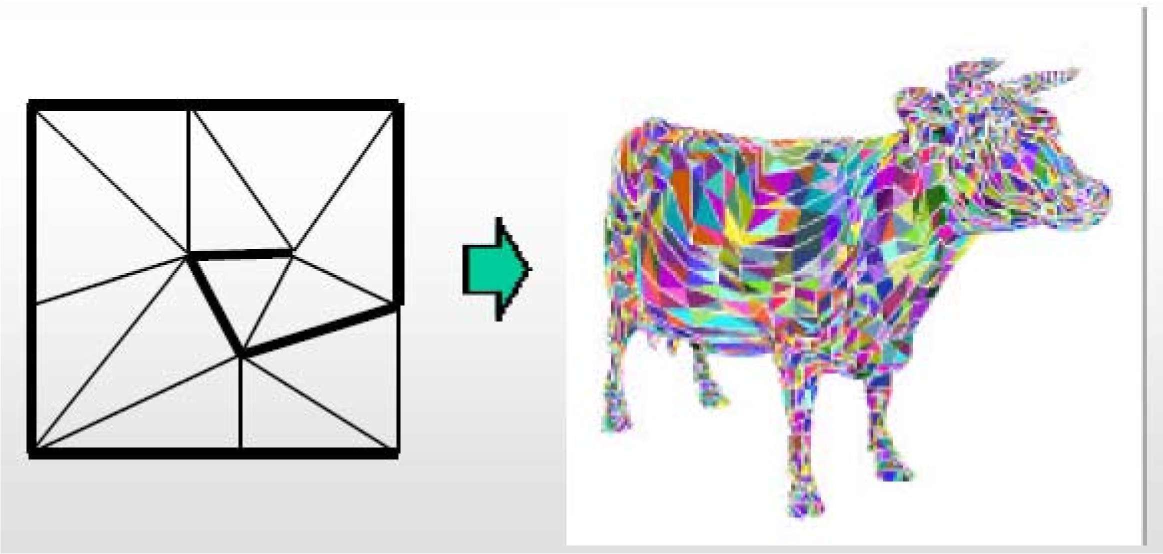
- In any planar graph edges incident on any vertex may be ordered consistently counter-clockwise





Mesh Travesal

- The mesh vertices may be ordered in a set of winding paths that traverse the mesh.

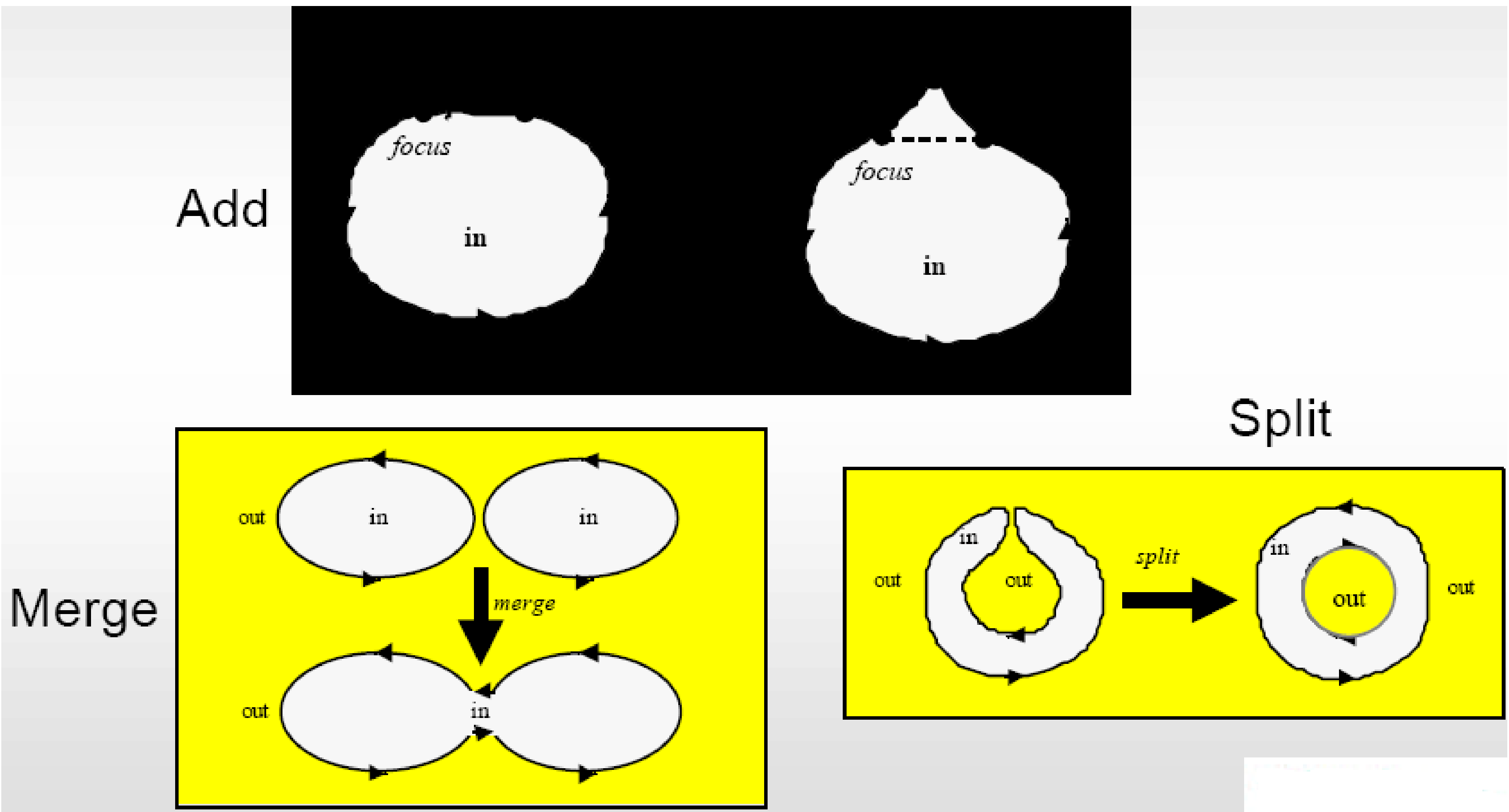


Code Words

- The topology may be encoded with :
  - add <degree>
  - split <offset>
  - merge <offset>

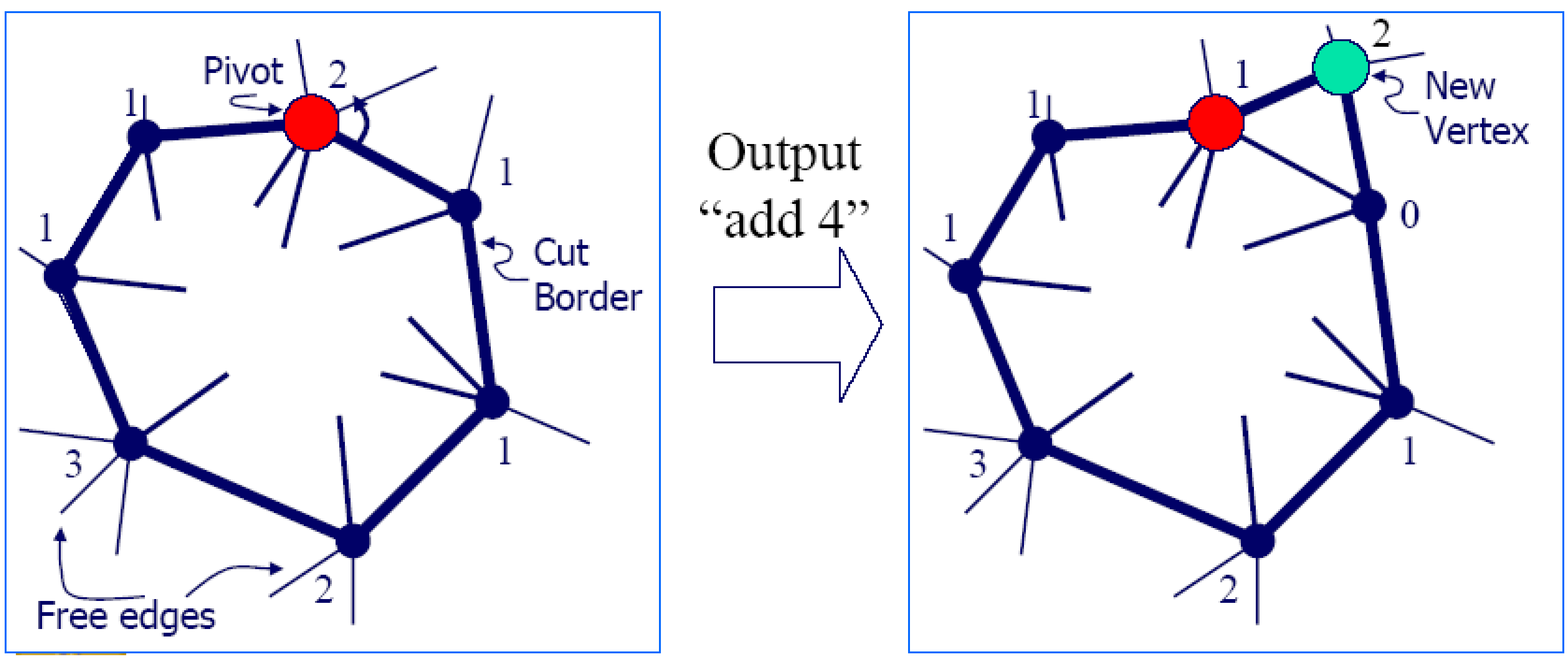
- and then entropy encoded (Huffman, run-length).

Topology Codewords

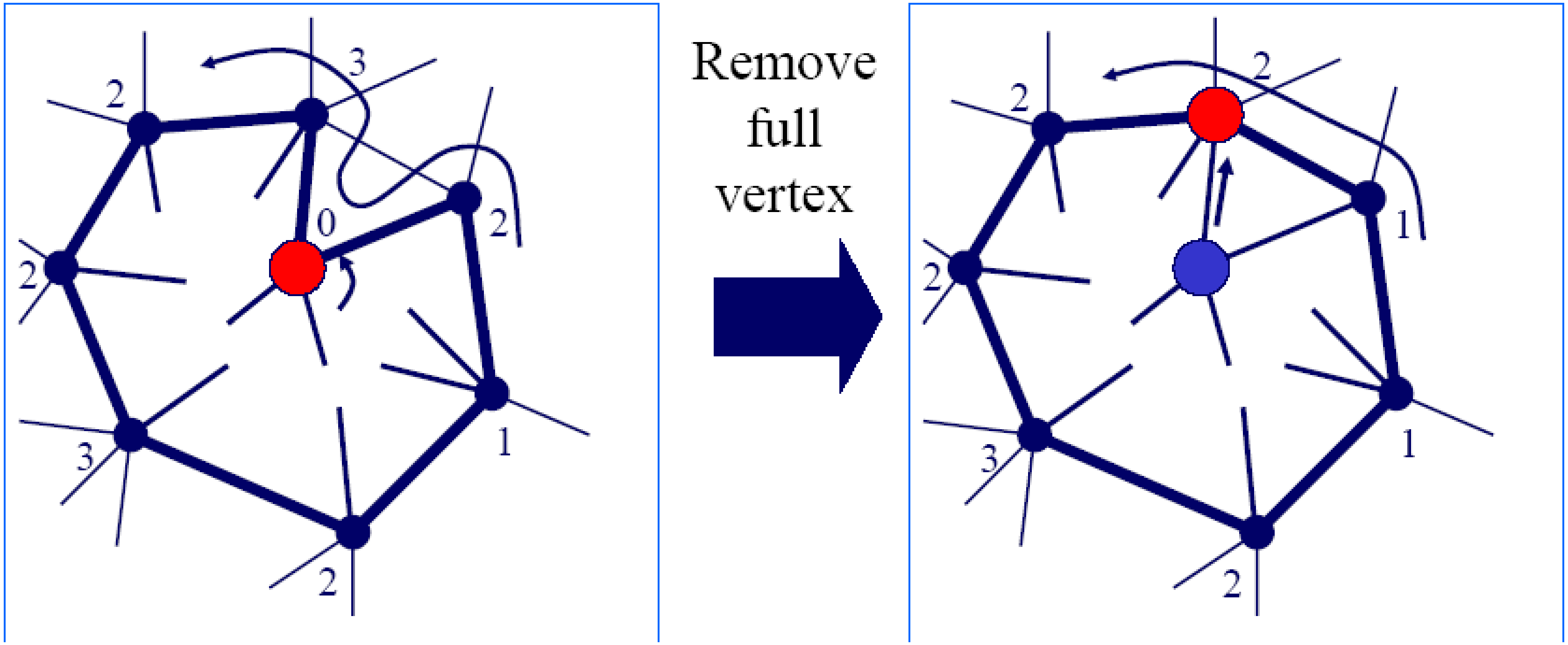


Topology Codewords

- Cut-border expansion: add <valence>

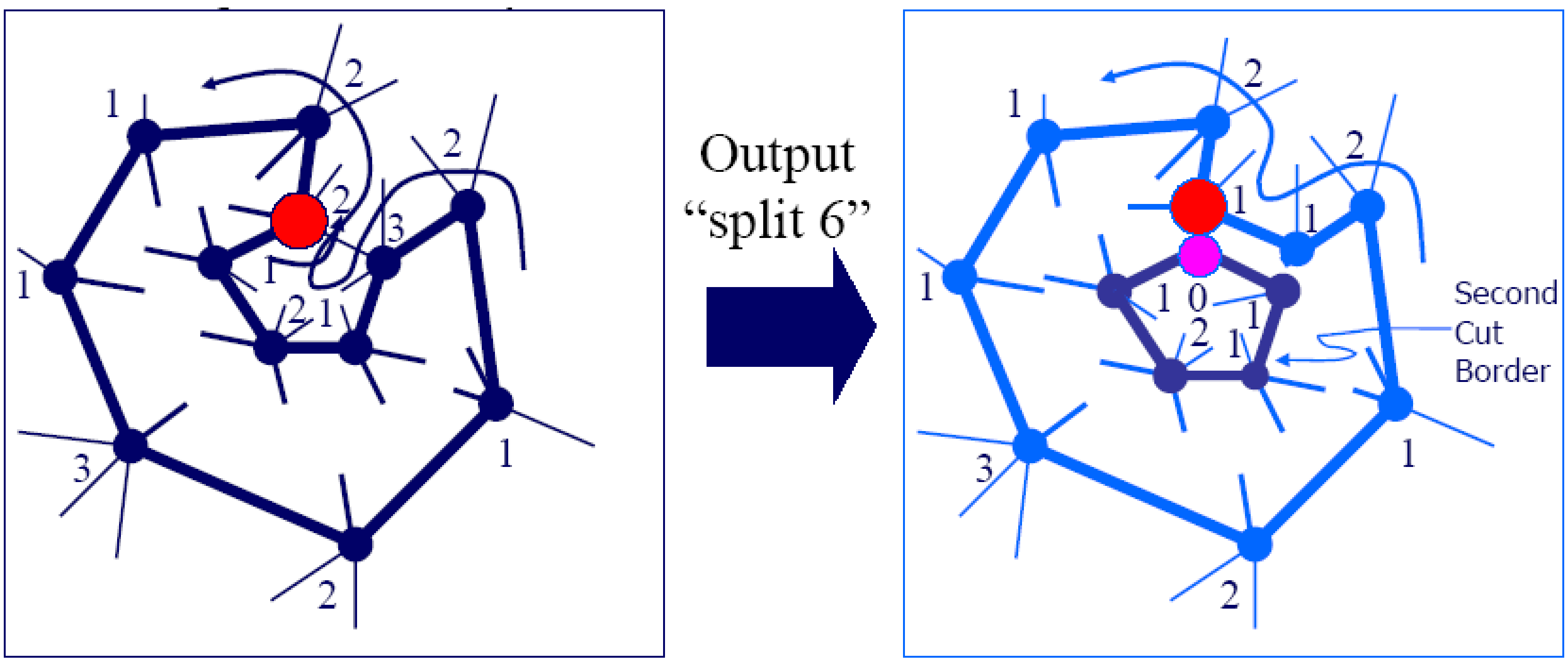


Topology Codewords

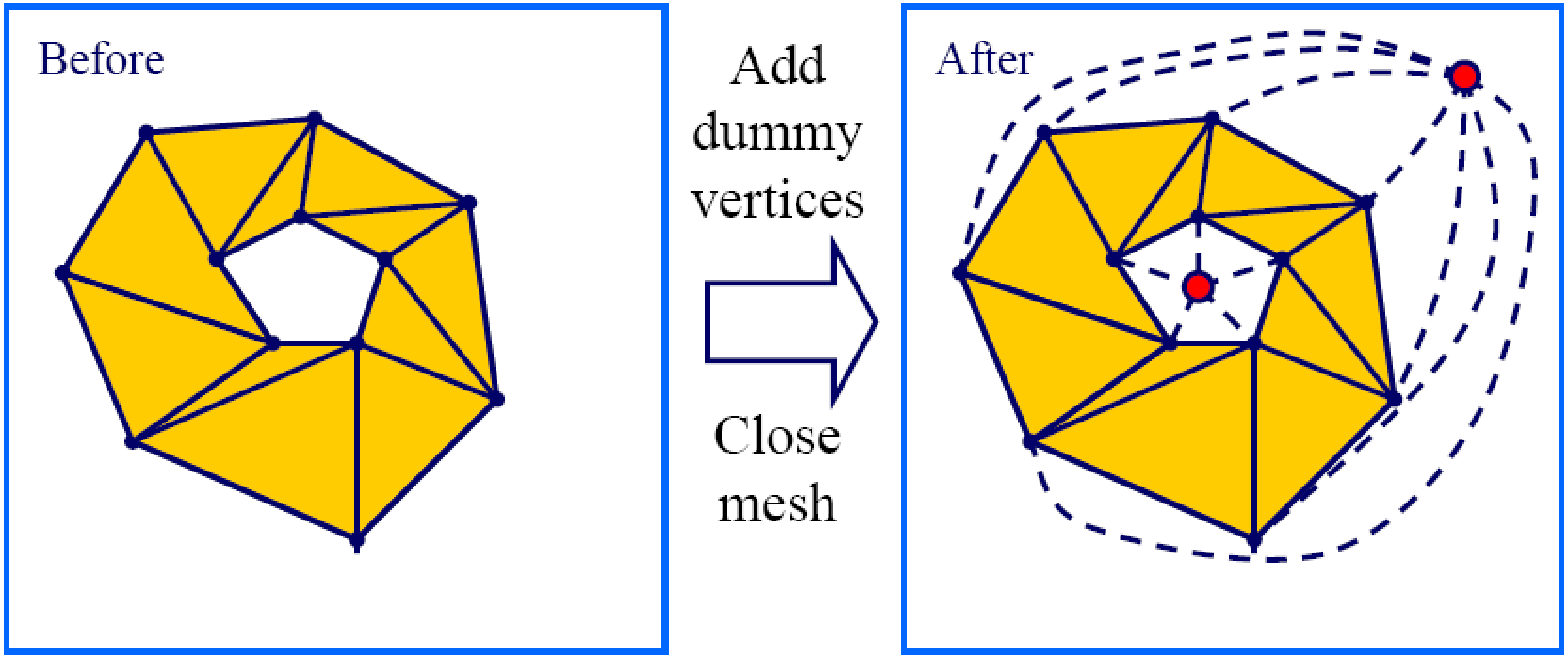


Topology Codewords

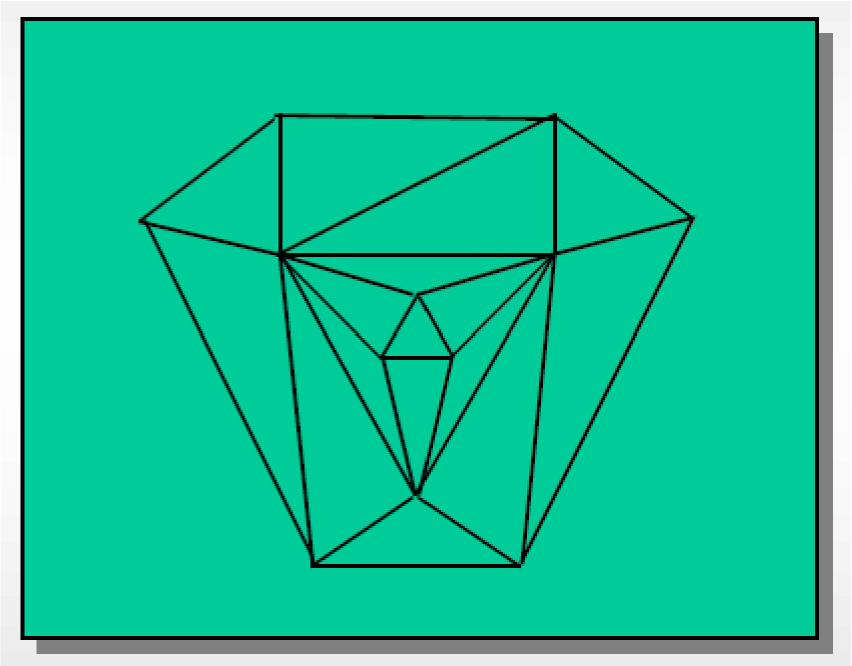
- split <offset>



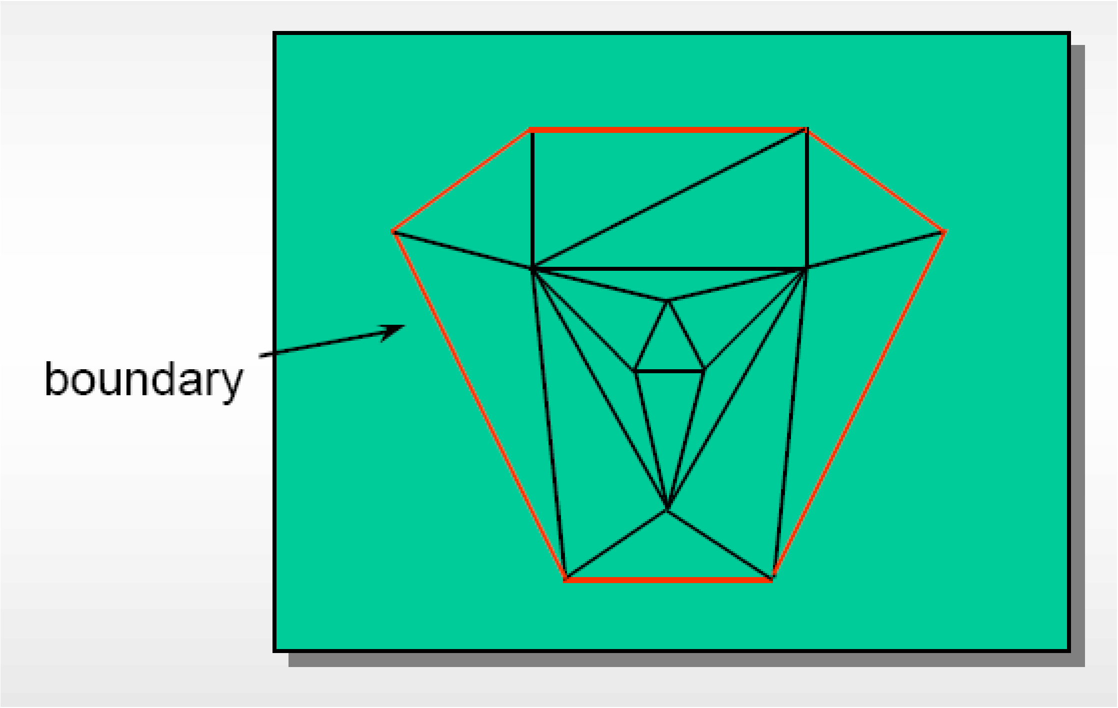
Topology Codewords



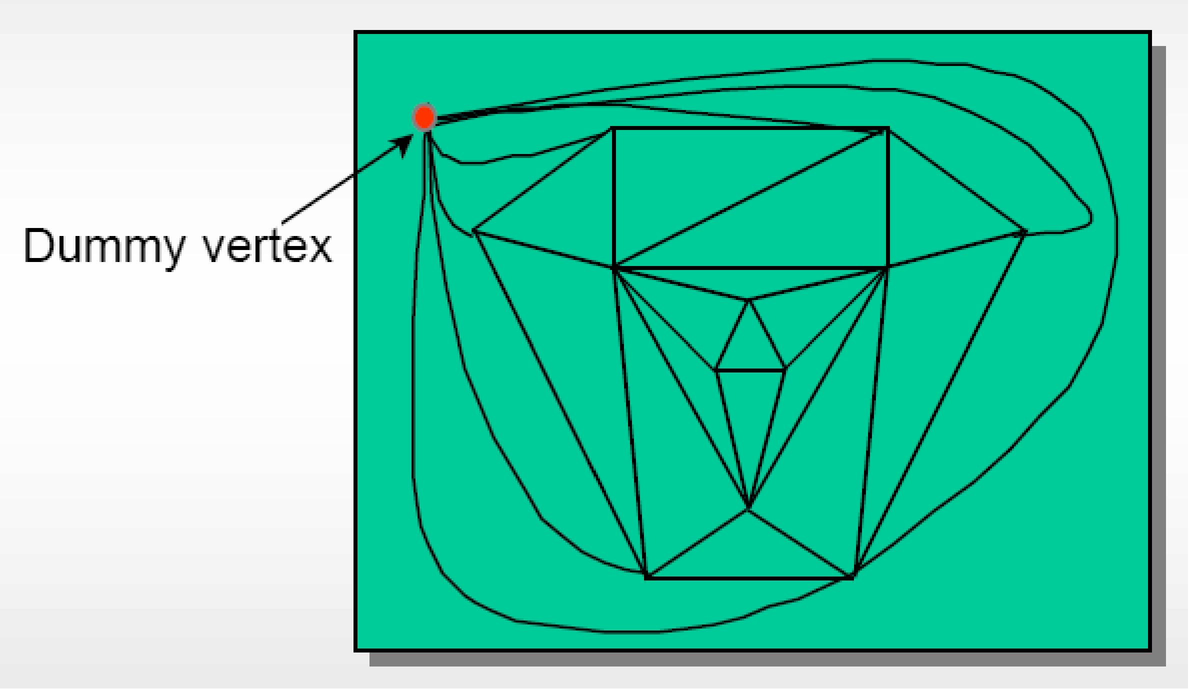
Example



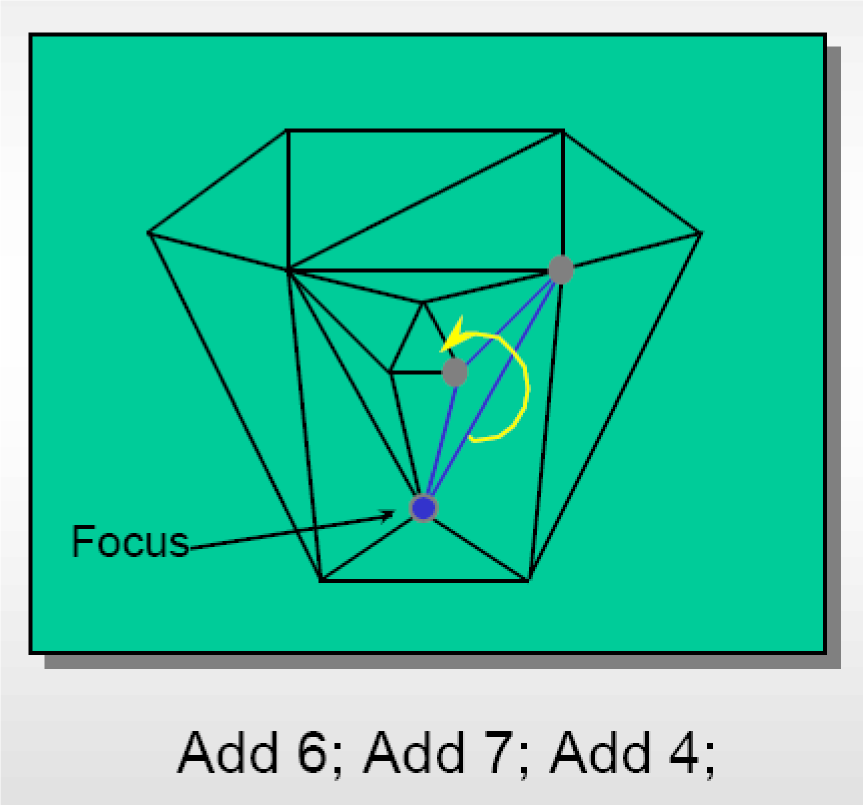
Encoding



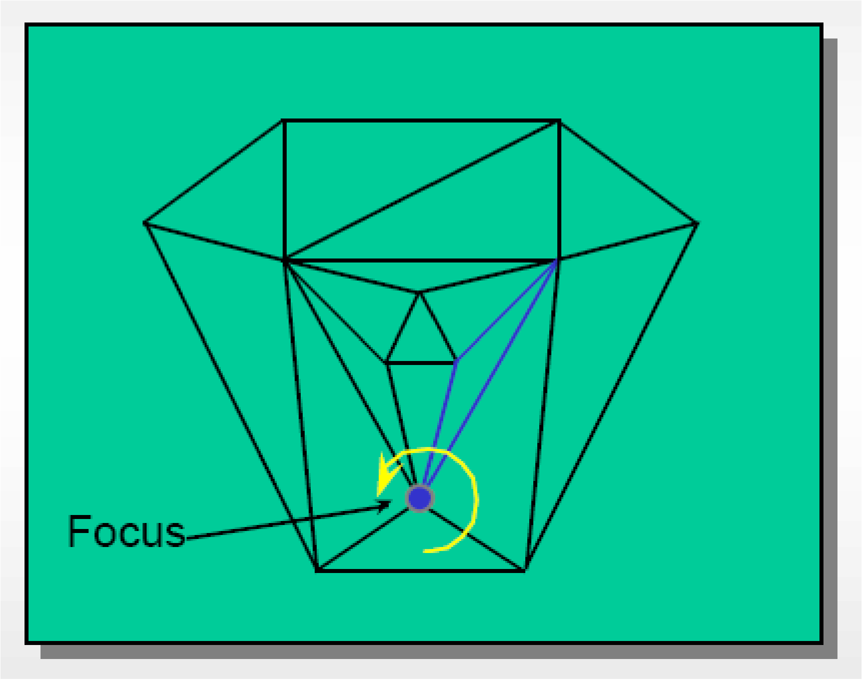
Encoding



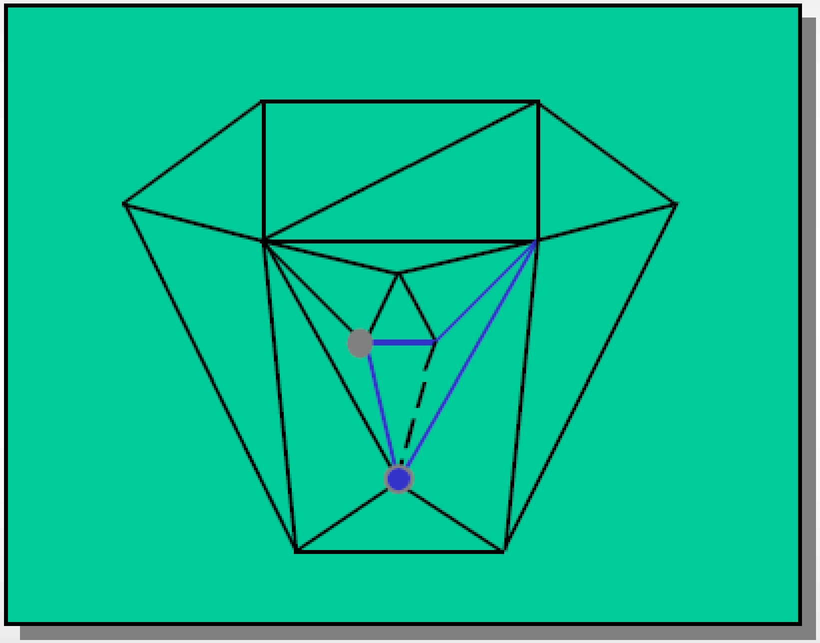
Encoding



Encoding

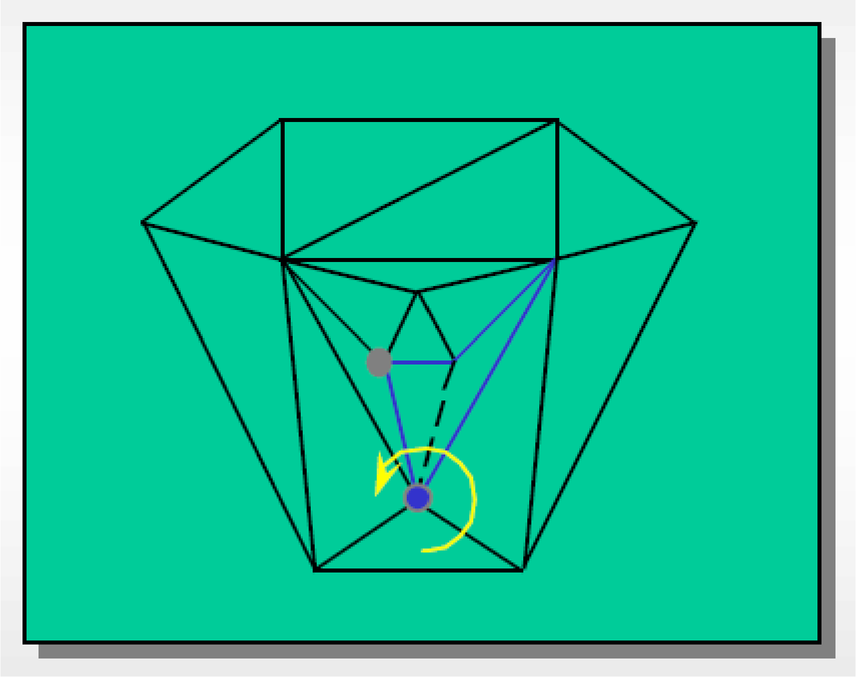


Encoding

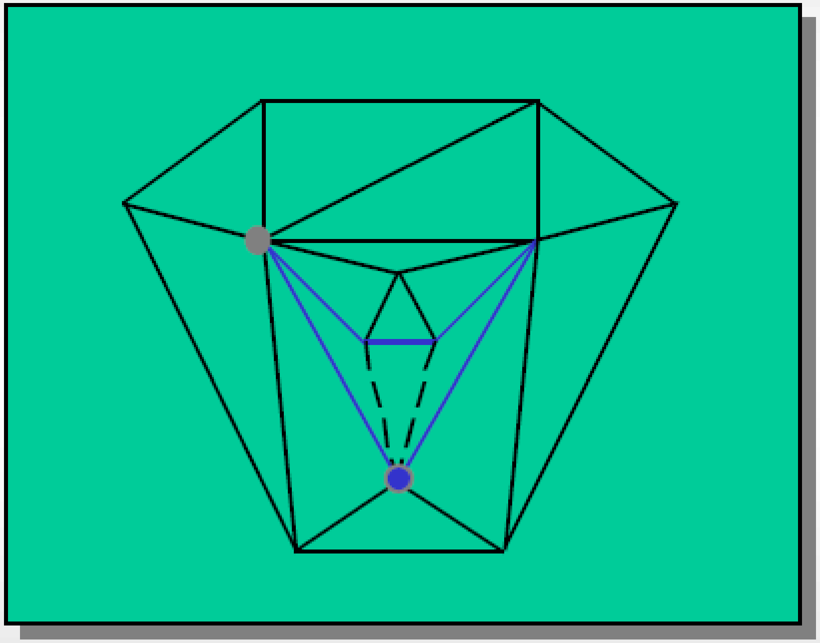


Add 4;

Encoding

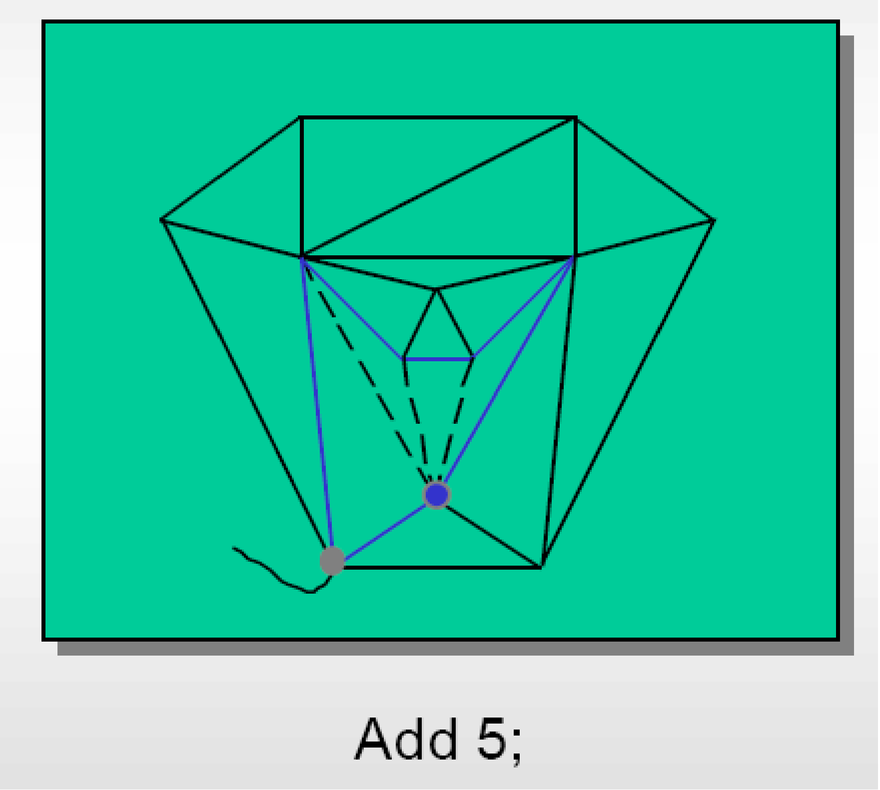


Encoding

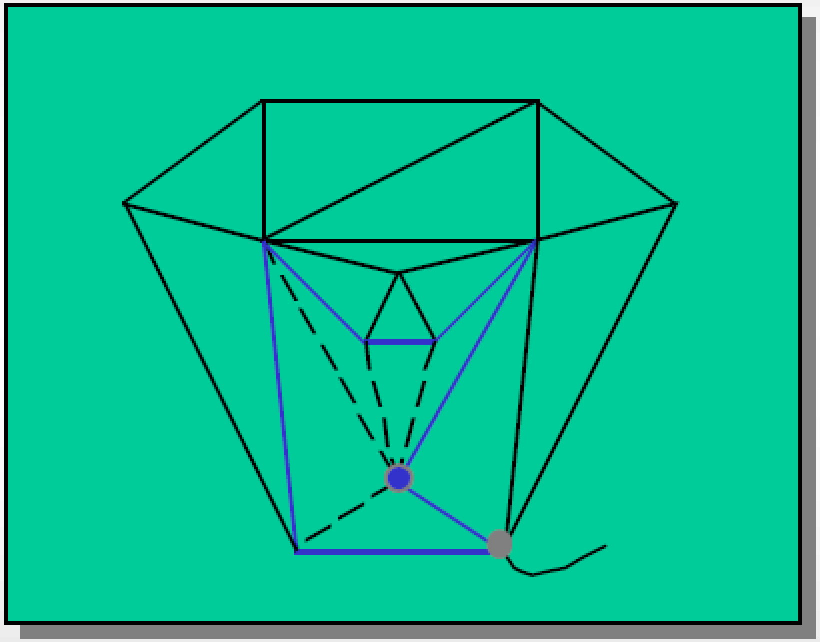


Add 8;

Encoding

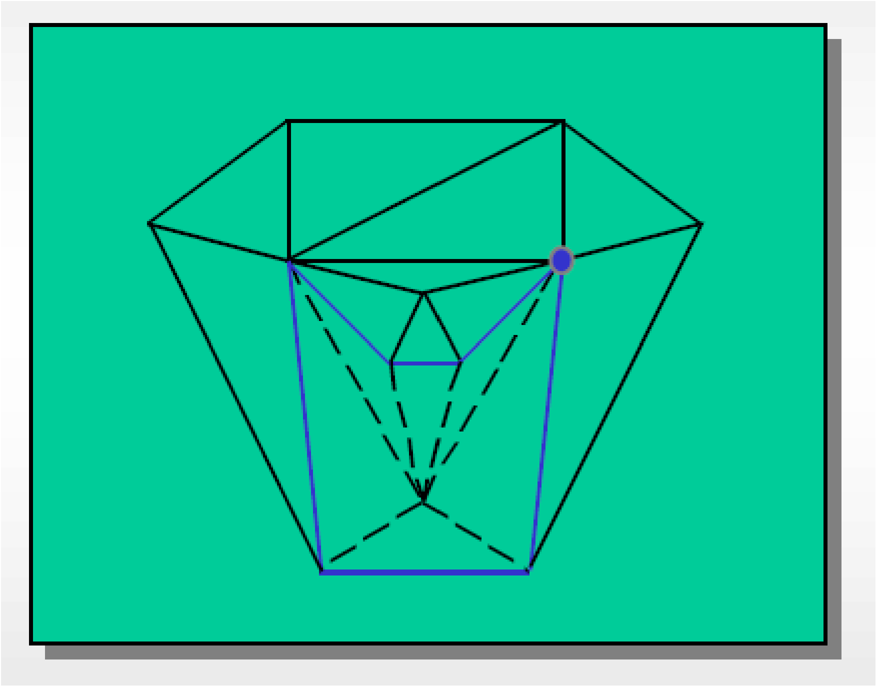


Encoding

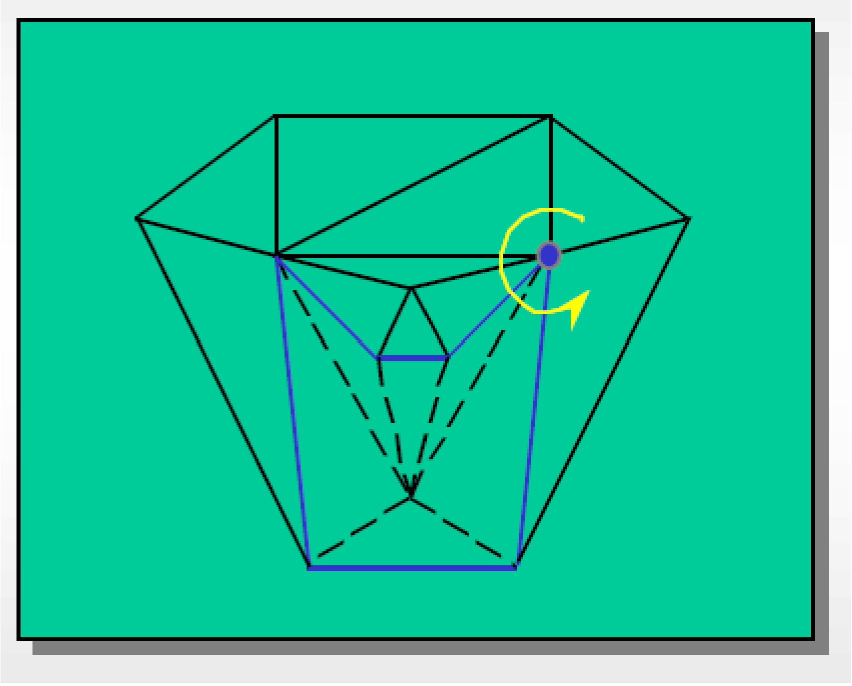


Add 5;
(focus full)

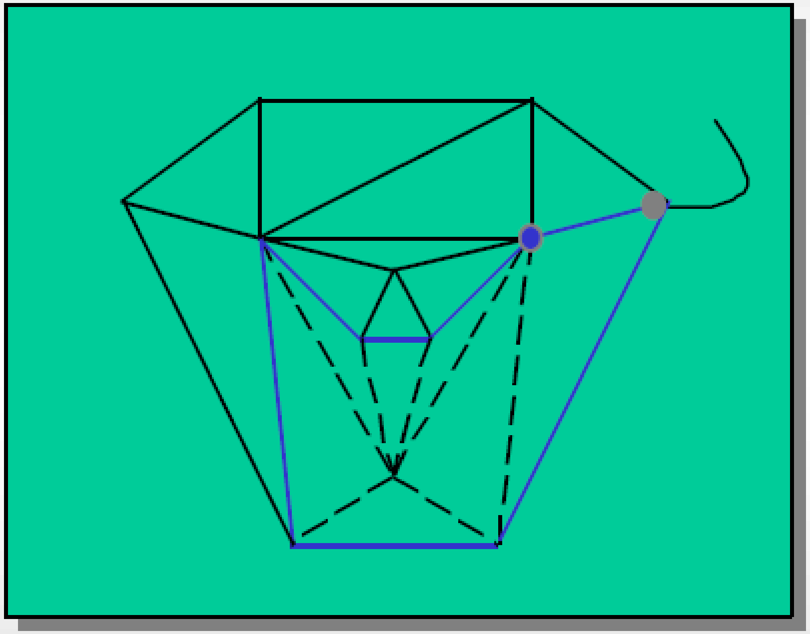
Encoding



Encoding

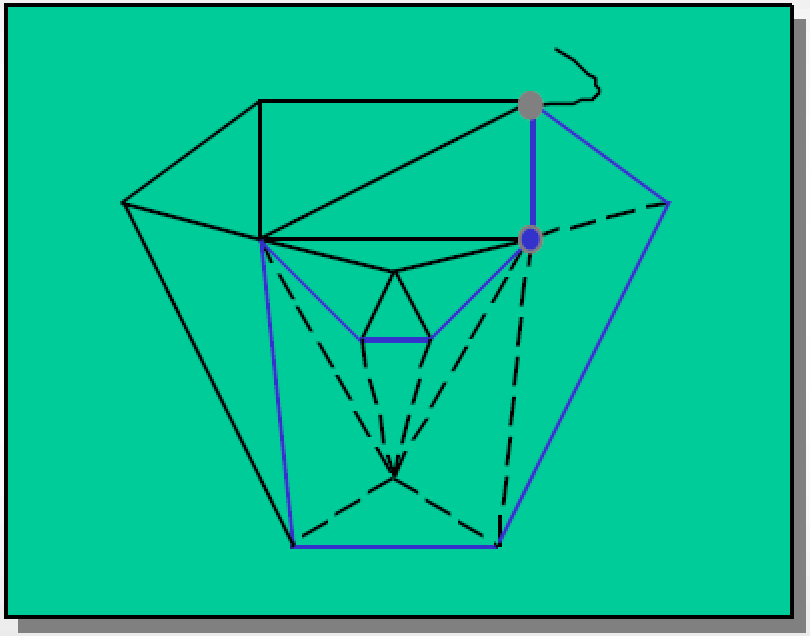


Encoding



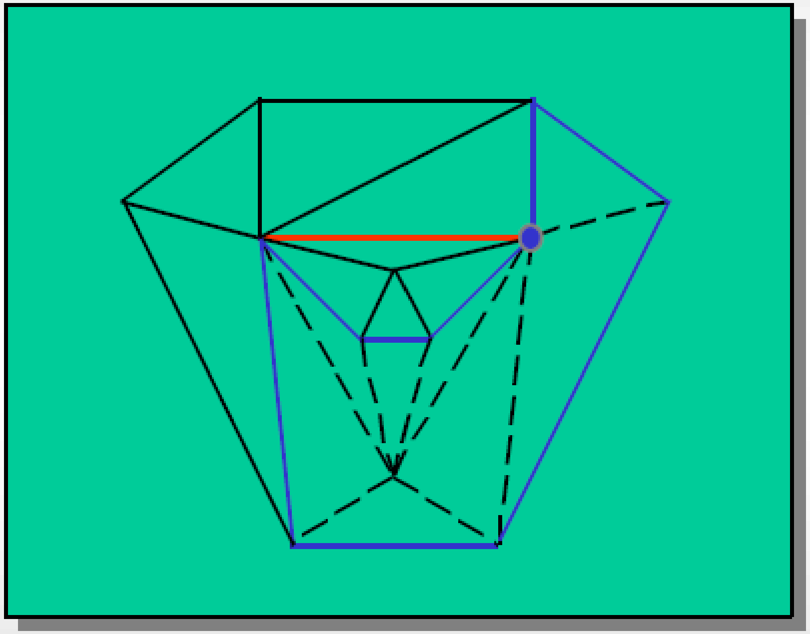
Add 4;

Encoding



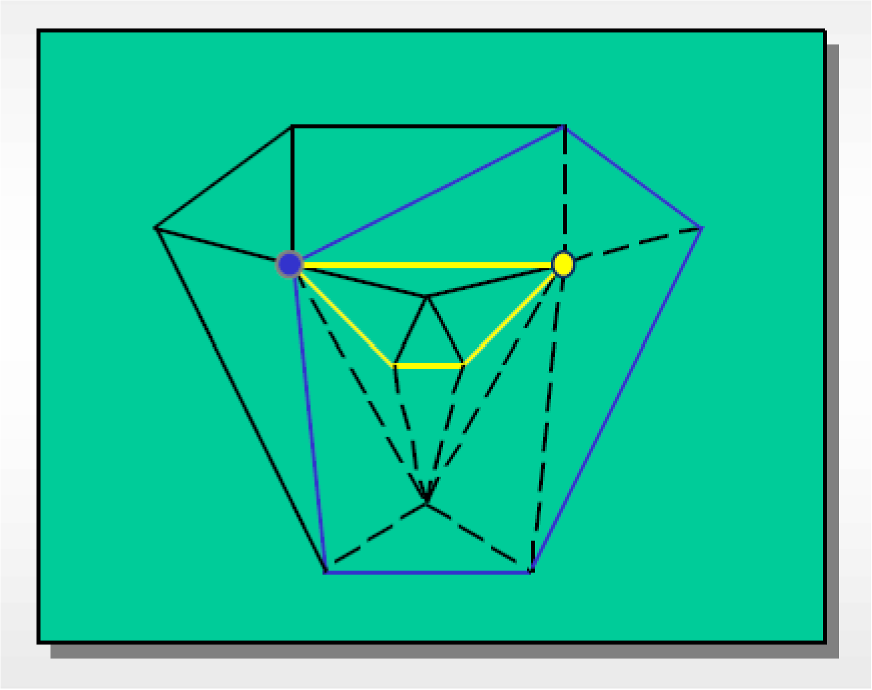
Add 5;

Encoding

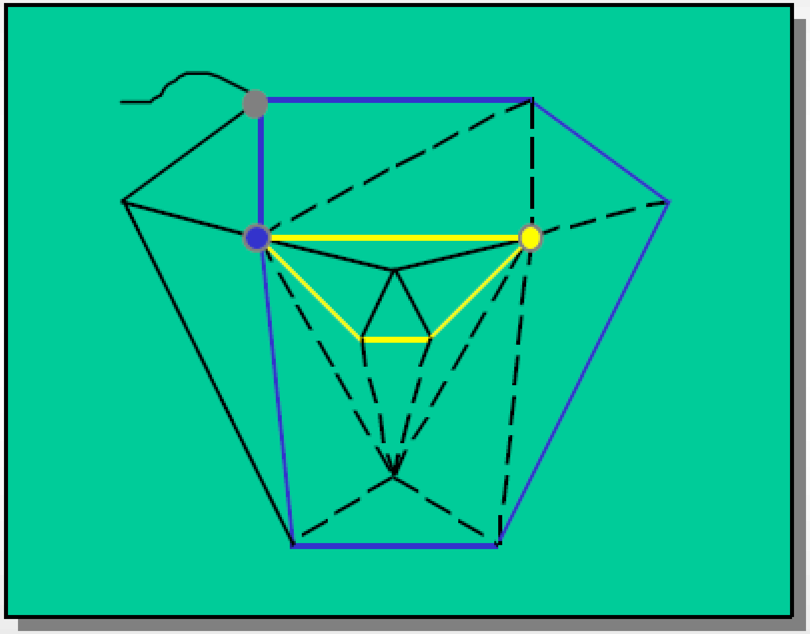


Split 5;

Encoding

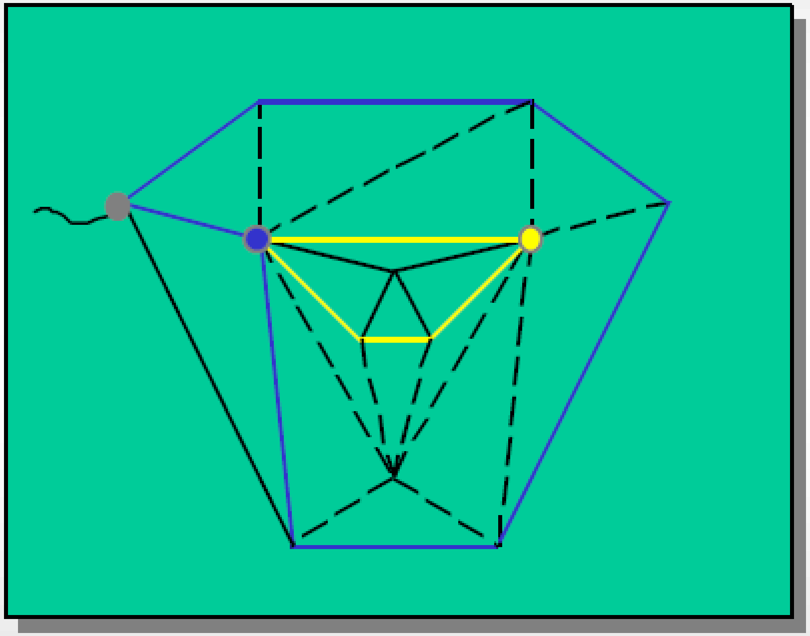


Encoding



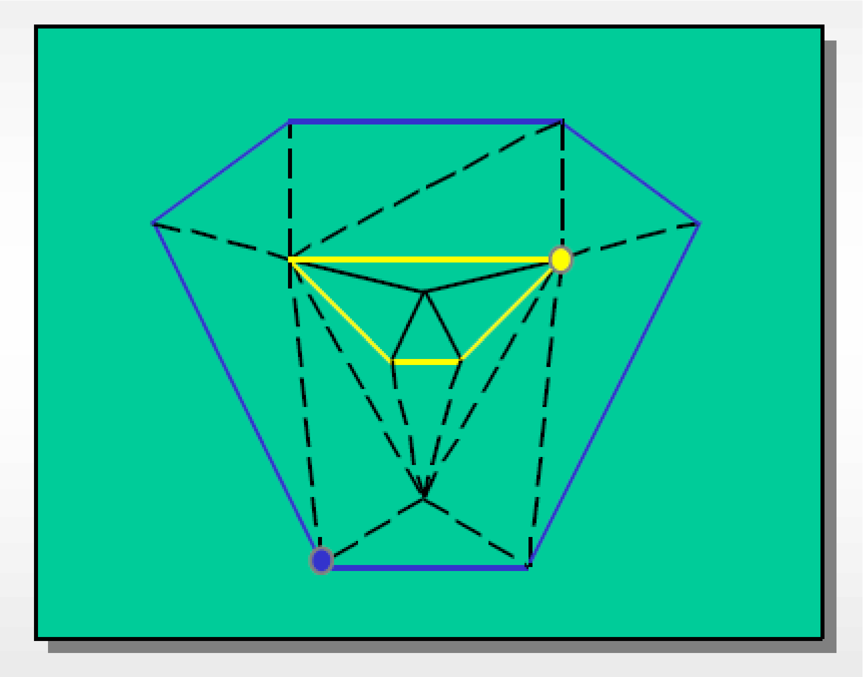
Add 4;

Encoding

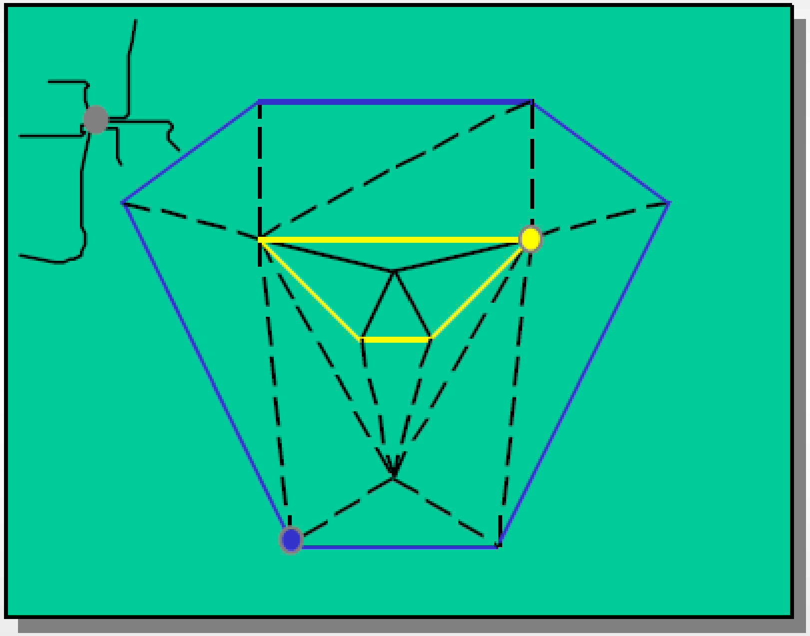


Add 4;
(focus full)

Encoding

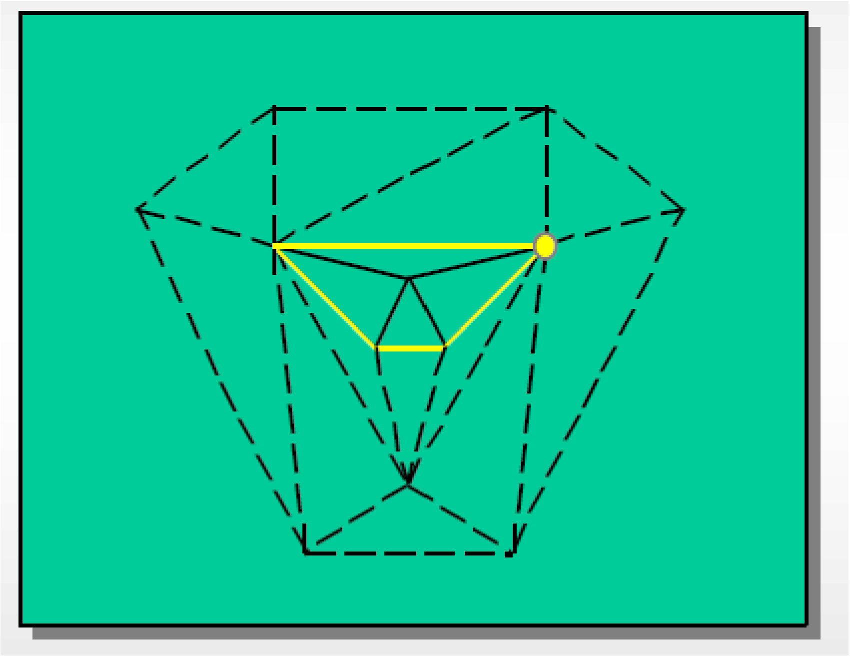


Encoding

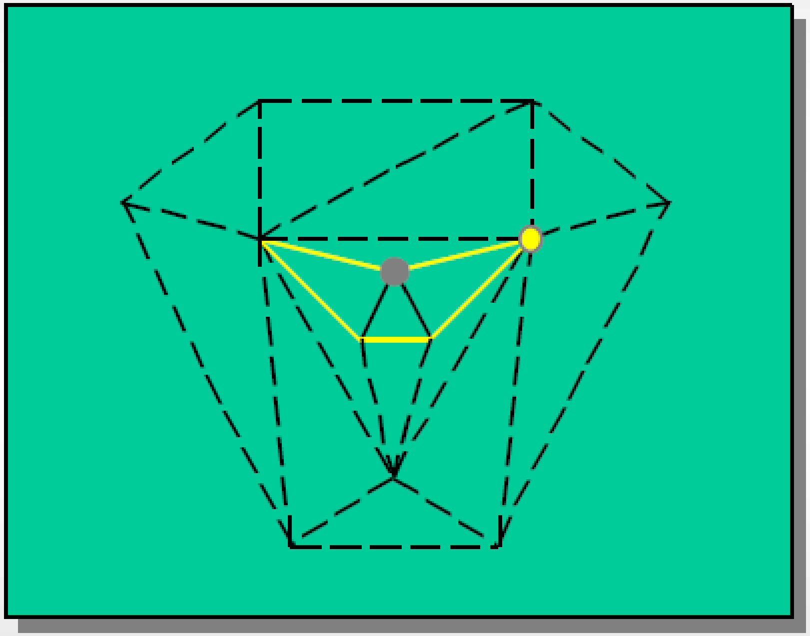


Add dummy 6;

Encoding

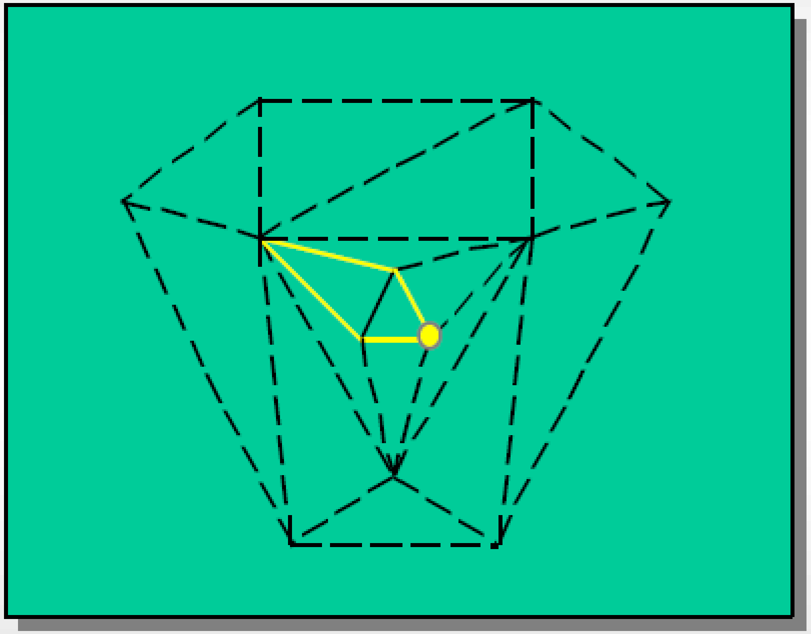


Encoding



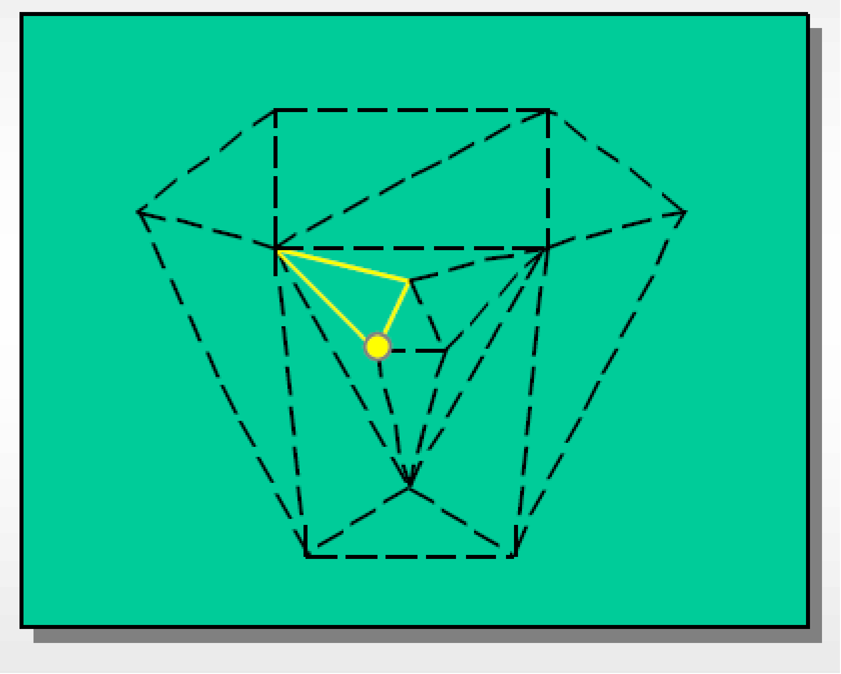
Add 4;
(focus full)

Encoding

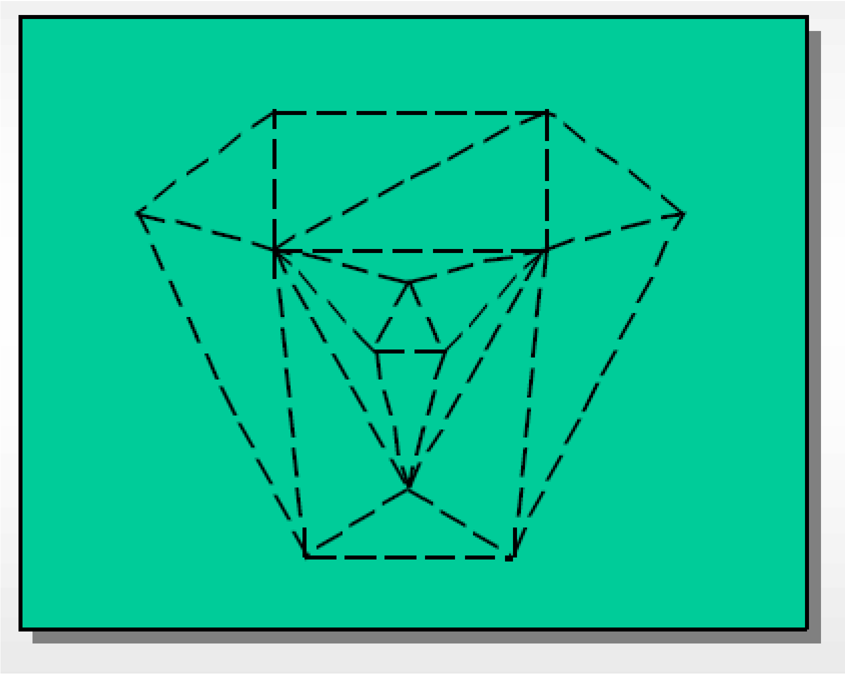


(focus full)

Encoding



Encoding

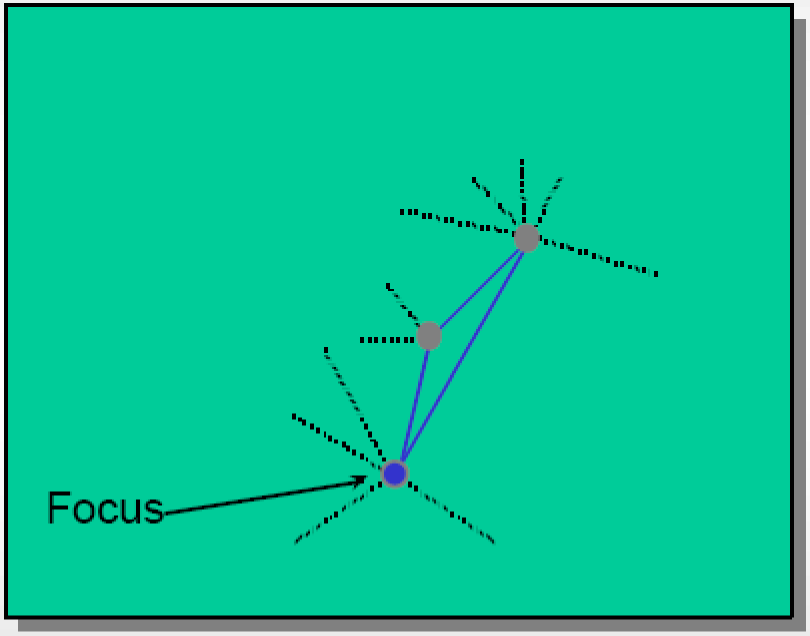


The Code

Add 6; Add 7; Add 4; Add 4; Add 8;
Add 5; Add 5; Add 4; Add 5; Split 5;
Add 4; Add 4; Add Dummy 6; Add 4;

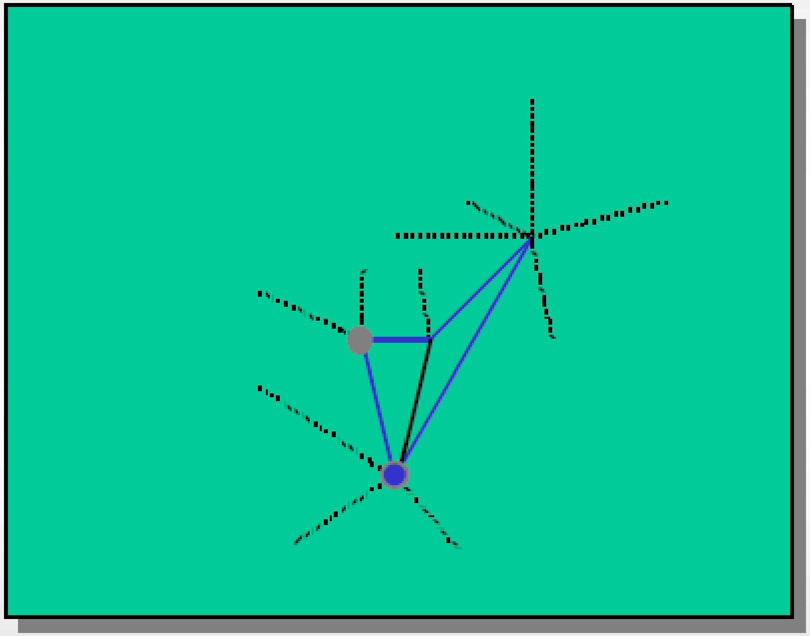
- For regular meshes (constant degree), spectacular compression ratios may be achieved.

Decoding



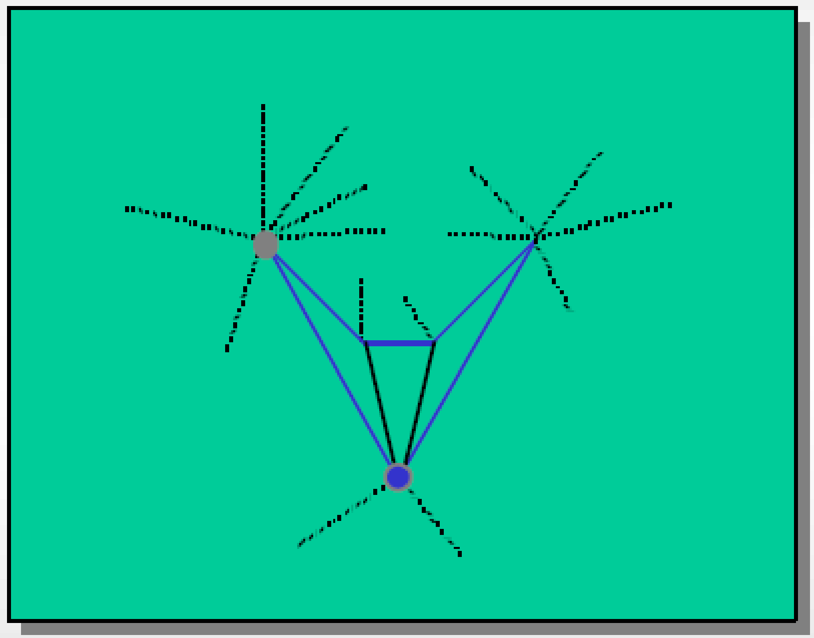
Add 6; Add 7; Add 4;

Decoding



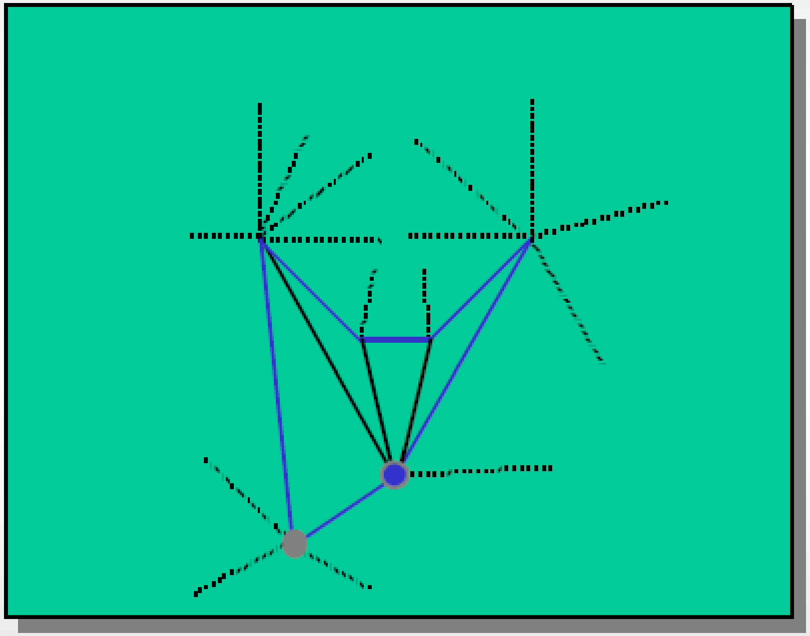
Add 4;

Decoding



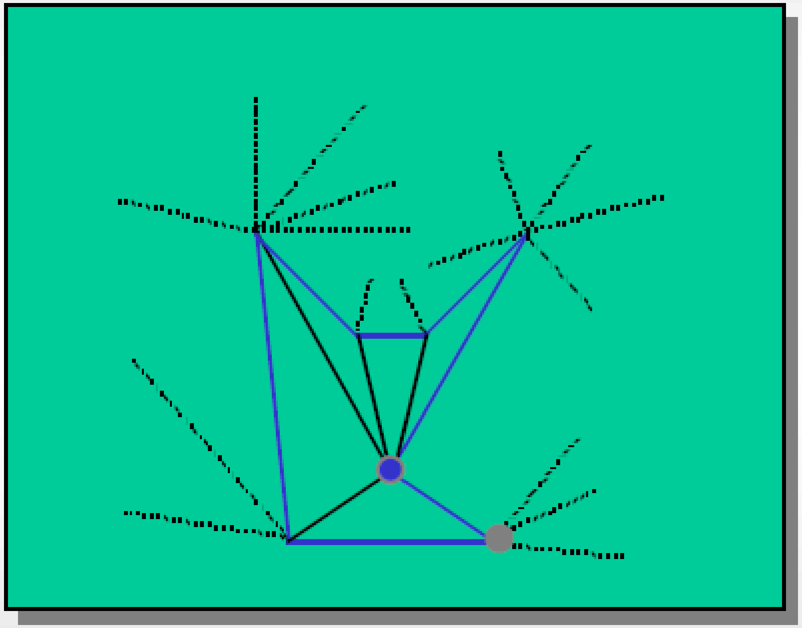
Add 8;

Decoding



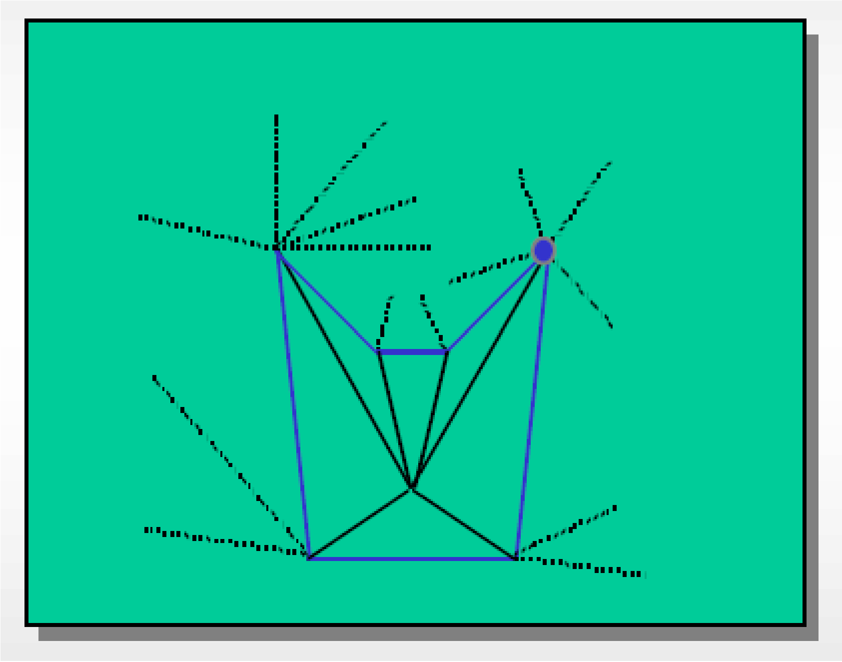
Add 5;

Decoding

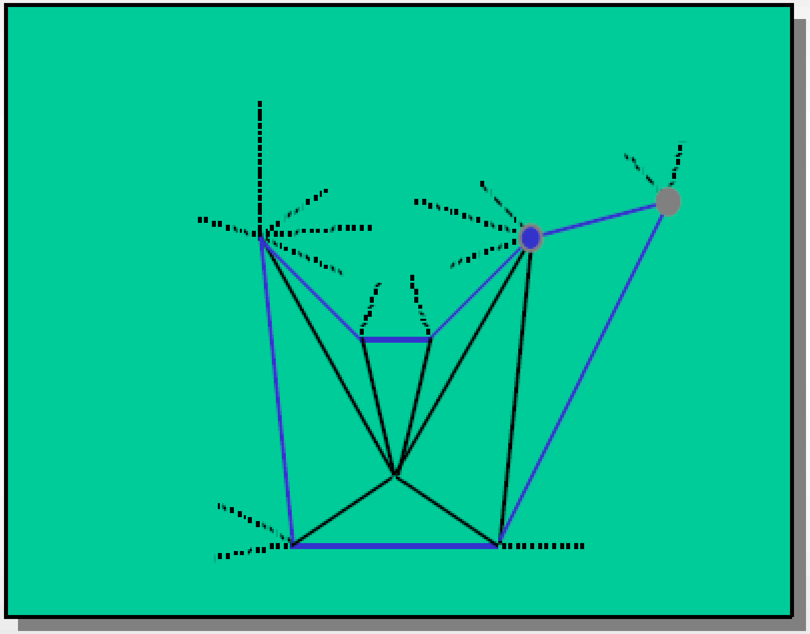


Add 5;
(focus full)

Decoding

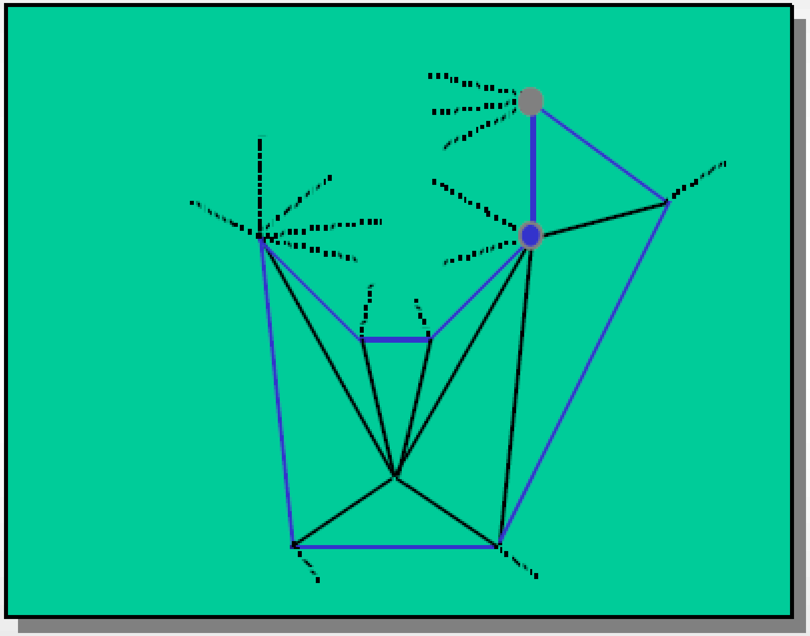


Decoding



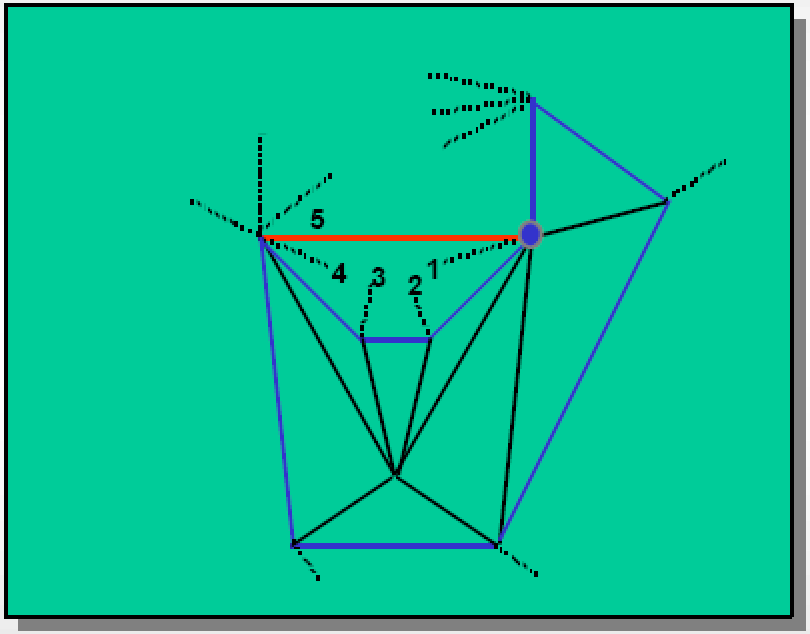
Add 4;

Decoding



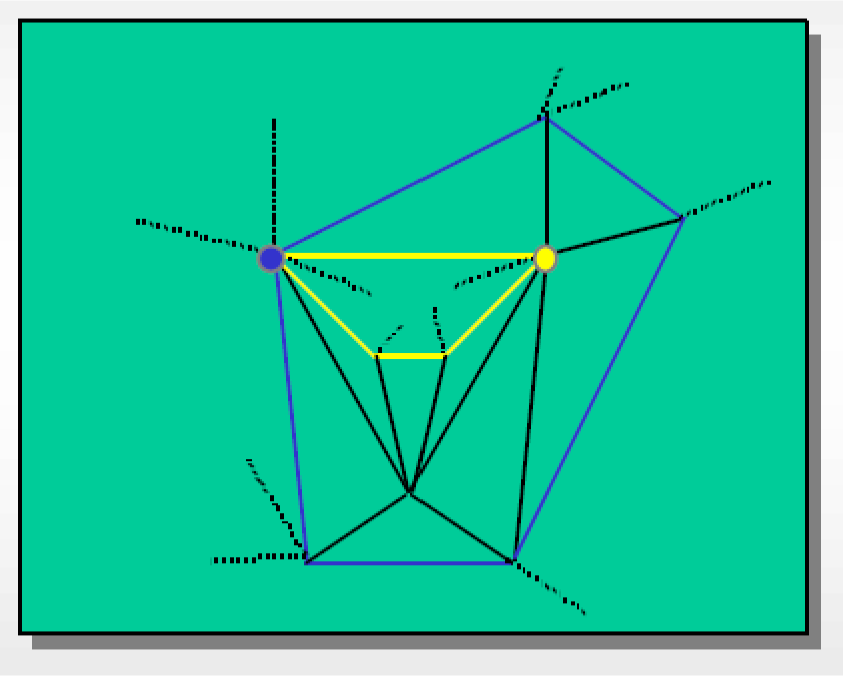
Add 5;

Decoding

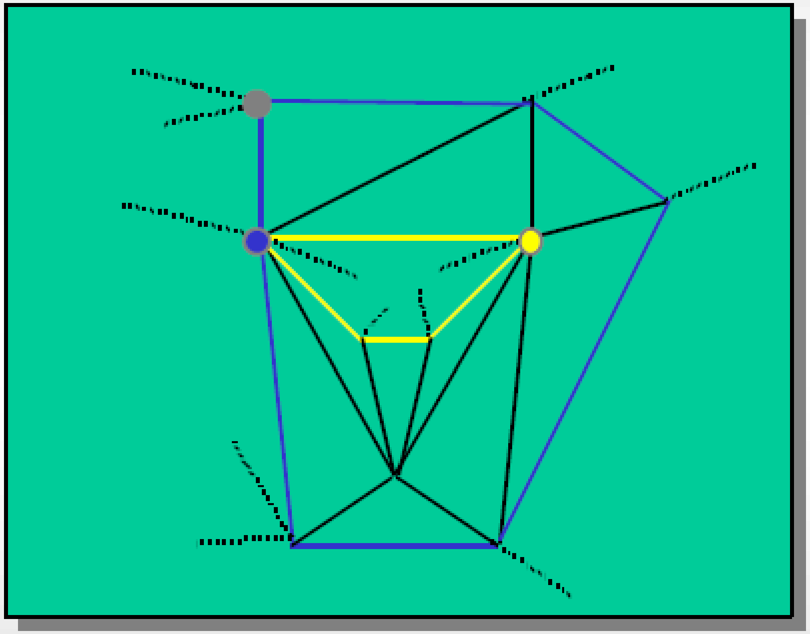


Split 5;
(focus full)

Decoding

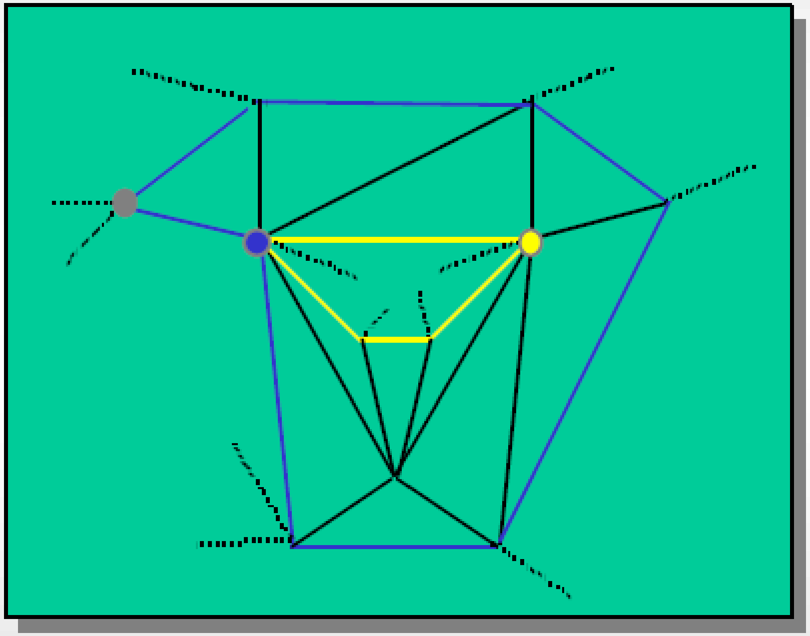


Decoding



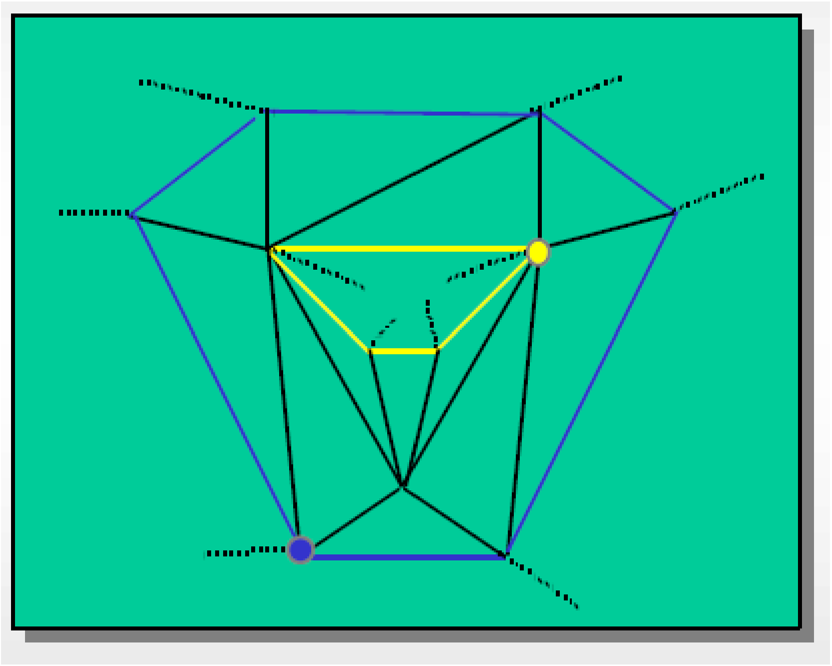
Add 4;

Decoding

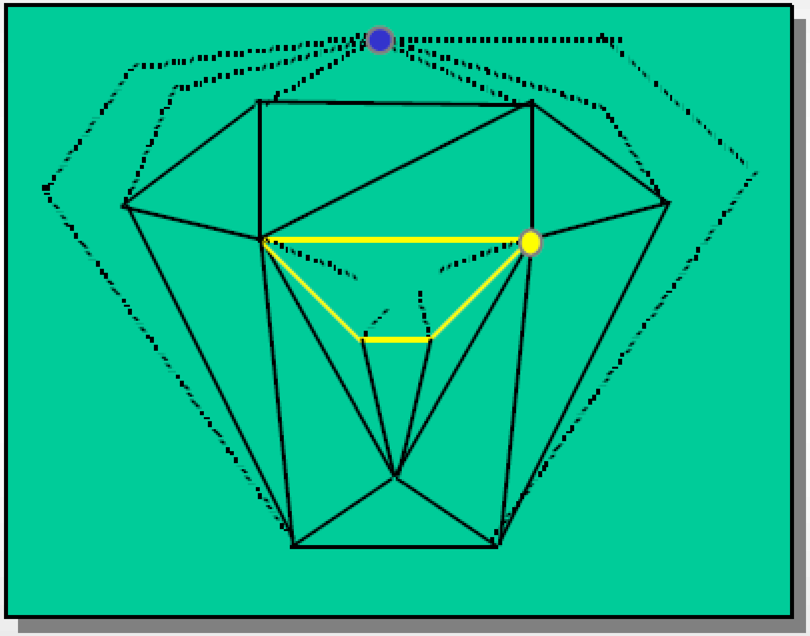


Add 4;
(focus full)

Decoding

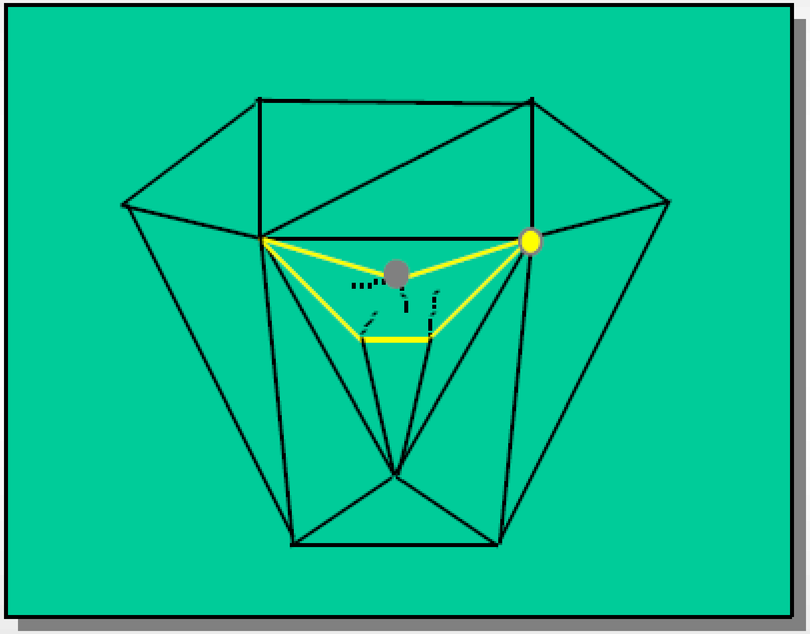


Decoding



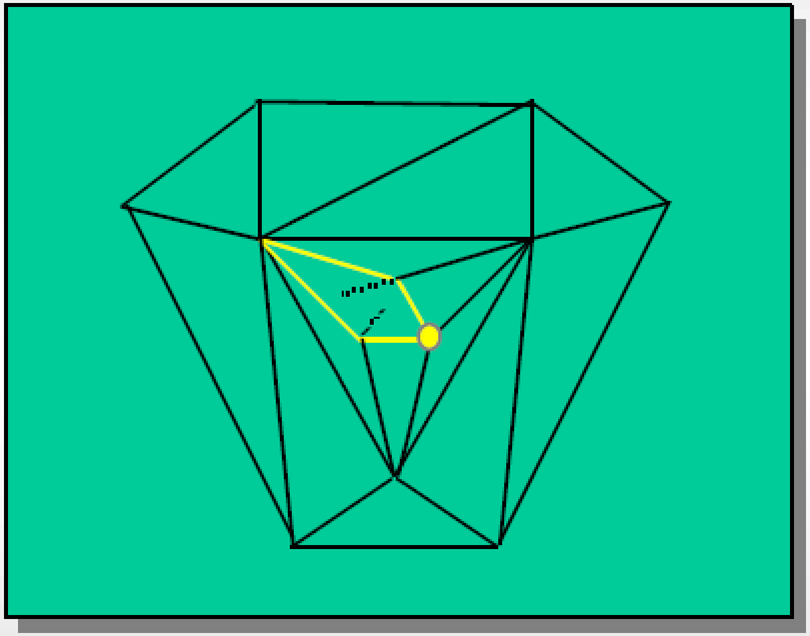
Add dummy 6;
(AL full - pop AL)

Decoding



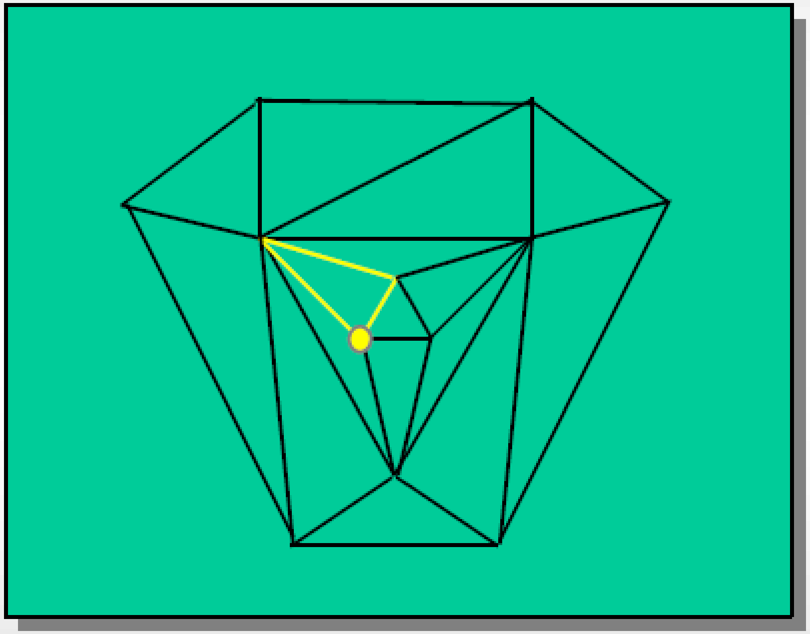
Add 4;
(focus full)

Decoding



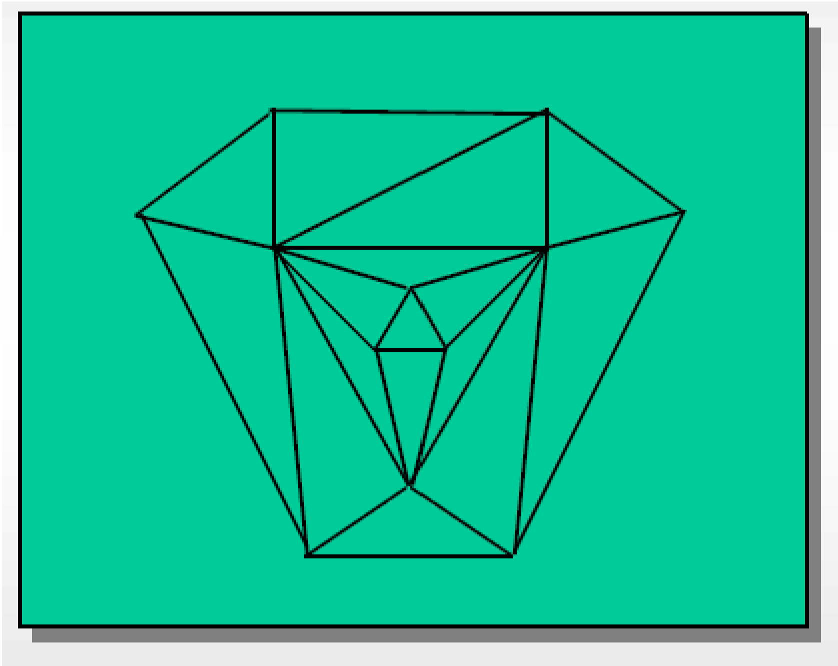
(focus full)

Decoding

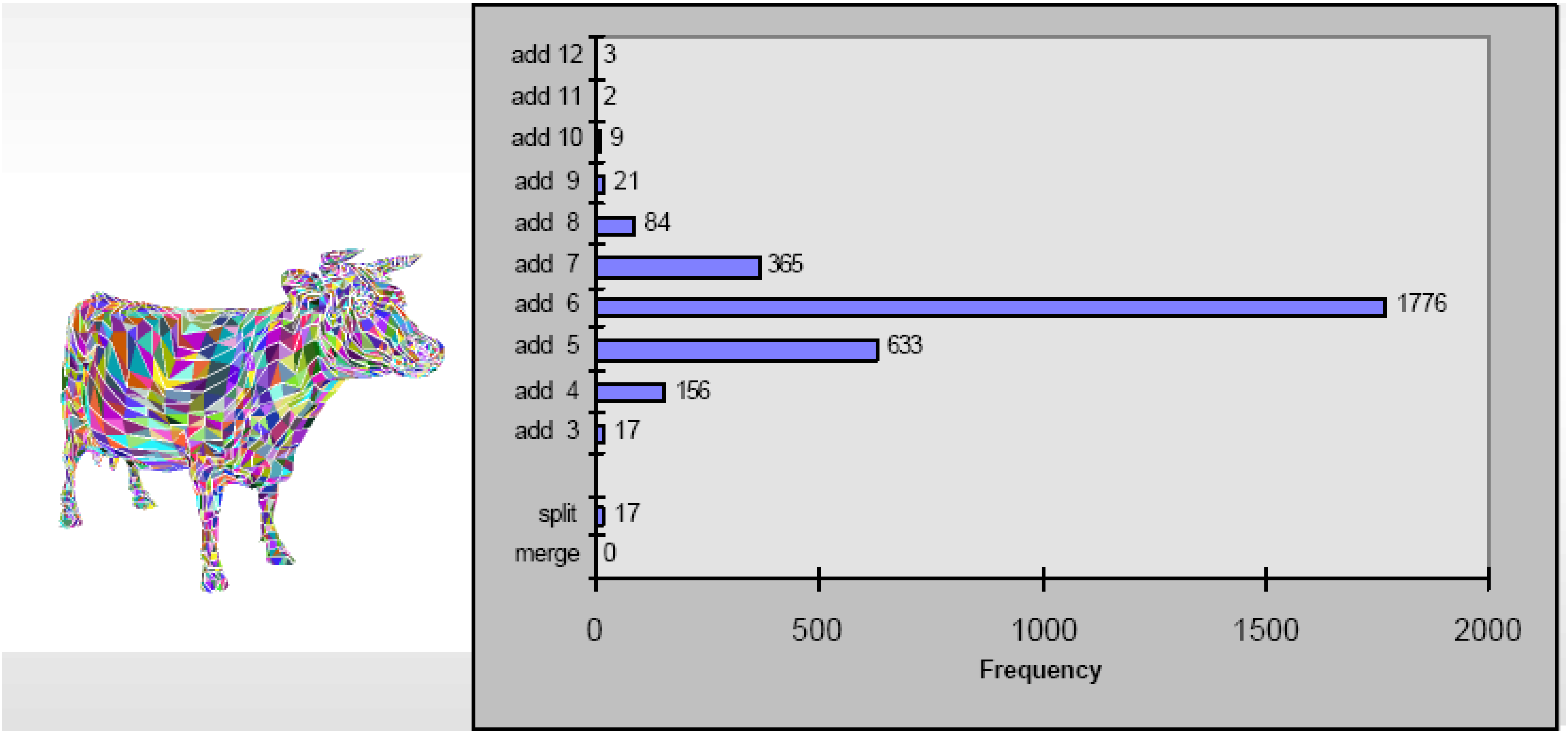


(focus full)
(AL full)

Decoding

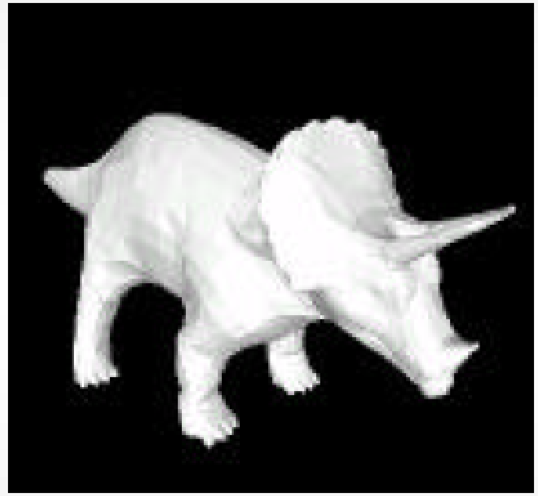


Example


add 12
3
add 11
2
add 10
9
add 9
21
add 8
84
add 7
365
add 6
1776
add 5
633
add 4
156
add 3
17
split
17
merge
0
0
500
1000
1500
2000
Frequency

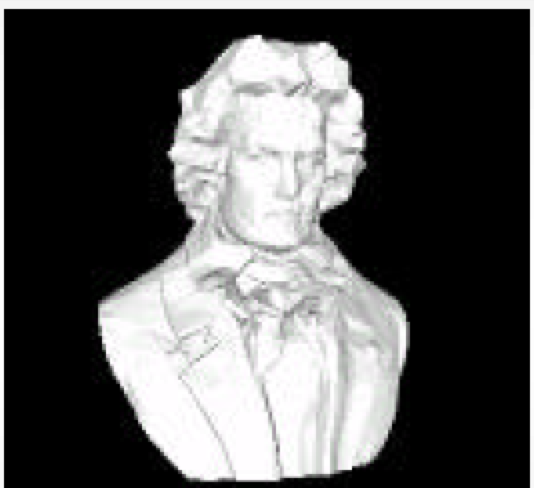
More Examples

Eight: 1,536 tri.



Triceratops: 5,660 tri.

Cow: 5,804 tri.



Beethoven: 5,028 tri.

Dodge: 16,646 tri.

Starship: 8,152 tri.

Results

Model	#tri.	bits/tri
Eight	1,536	0.2
Triceratops	5,666	1.4
Cow	5,804	1.1
Beethoven	5,028	1.4
Dodge	16,646	0.9
Starship	8,152	0.5
Average		**0.9**

Performance

- Disadvantages:
  - No theoretical upper bound on code length
- Advantages:
  - Gives **very good** compression rates (approx 2 bits/vertex) on typical meshes
  - Gives excellent rates on highly regular meshes

Extensions

- Merge operation required when genus > 0
  - Occurs when two different cut-borders intersect
- Non-manifolds treated by cutting into manifold pieces

Several Other Solutions

- **Deering: Generalized triangle strips**
  - Use buffer to avoid sending vertices more than once
  - Designed for hardware decompression
- **Taubin&Rossignac: Topological Surgery**
  - Efficient encoding of vertex and triangle trees
  - MPEG-4 Standard
- **Gumhold&Strasser: Cutborder**
  - Encode spiraling pattern and offsets that define bifurcations
- **Touma&Gotsman**
  - Encode vertex valence and bifurcation offsets (great for regular meshes)
- **Rossignac&Szymczak&King: Edgebreaker**
  - No need to encode offsets of spiraling pattern
  - 1.83T bits **guaranteed**, 1.0T bits demonstrated for large models

Discussions

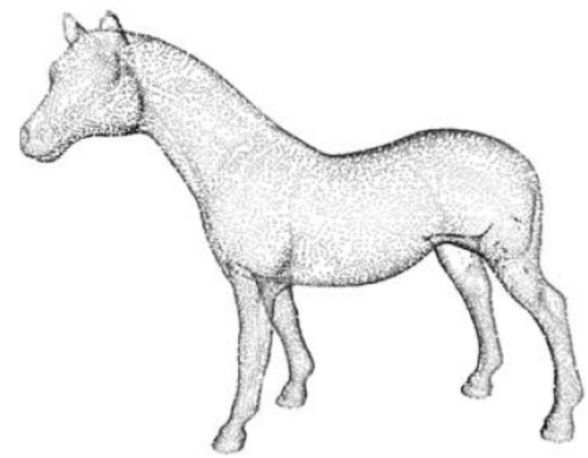
Geometry Encoding

Vertex Data

- **Position: x y z**
- **Normal: nx ny nz**
- **Color: r g b {a}**
- **Texture coordinates: s t {r} {q}**
- **(Others)**

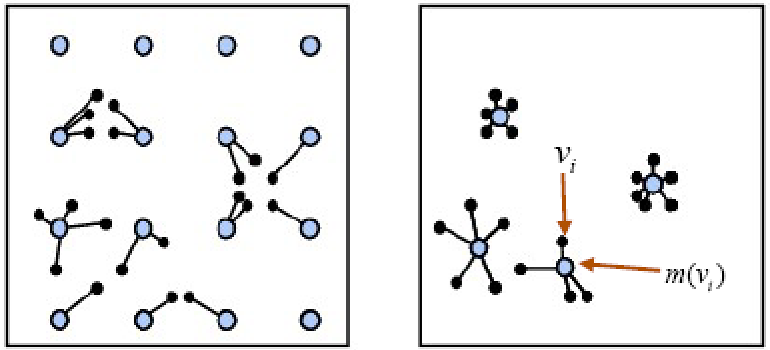
The Geometry

- Vertex coordinates (x, y, z) are
  - Floating point values
  - Almost unrestricted in:
    - range
    - precision
  - **Uniformly** spread in 3D
- Compression exploits input redundancy
  - **hard to find in raw geometry data**
- **Lossy compression is OK!!**



Quantization

- Map n values v_i to $k<<n$ values $m(v_i)$, without losing **too much** information
- Quantization error: $Err(v,m) = \sum_{i=1}^{n} \left\| v_i - m(v_i) \right\|^2$
- Find k and m such that $Err(v,m)$ is minimized



Uniform   Non-uniform

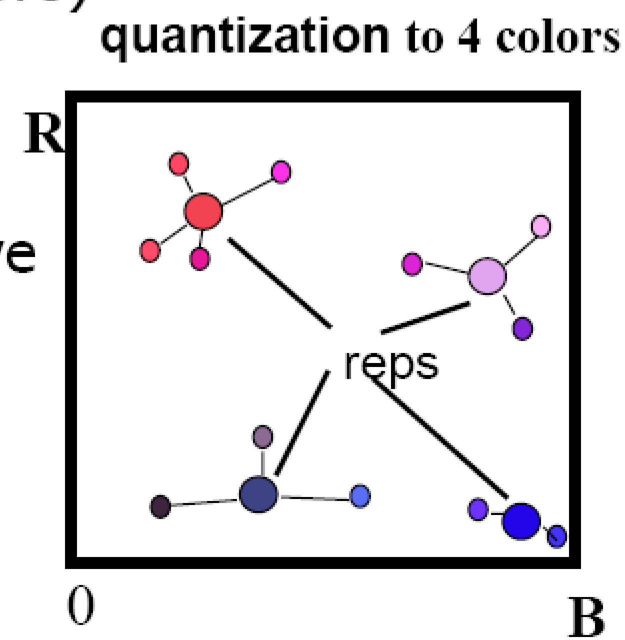
Quantization

- Example: rounding a set of doubles into integers

- Applications:
  - Image and voice compression
  - Voice recognition
  - Color display
  - Geometric compression

Example: color quantization

- Used for limited dynamic-range displays (e.g. an 8 bit display can display only 256 different colors)
- Reducing number of colors
  - Choosing set of representative colors (*colormap* or *palette*)
  - Map rest of colors to them
- Usually uses 256 colors



Representatives

- How to choose representative colors?
  – Fixed representatives, image independent - fast
  – Image content dependent – slow
- Which image colors mapped to which representatives?
  – Nearest representative - slow
  – By space partitioning - fast

Color quantization examples

256 colors

64 colors

16 colors

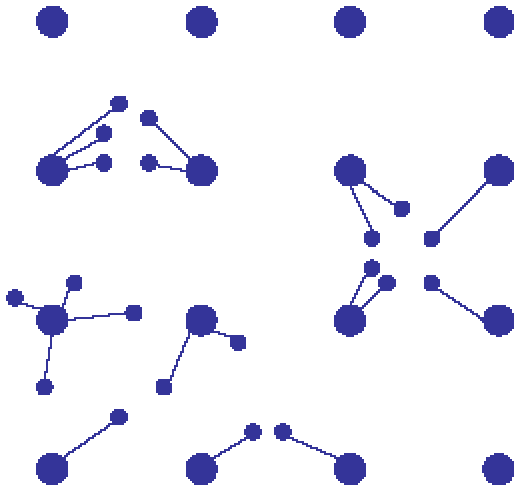
8 colors

4 colors

2 colors

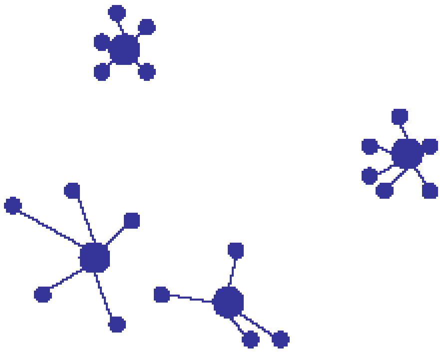
Uniform Quantization

- Quantization space partitioned into equal sized regions (e.g. grid) – fixed representatives
- Input independent
- Some representatives may be *wasted*
- Common way for 24->8 bit color quantization: retain 3+3+2 most significant bits of R, G & B components



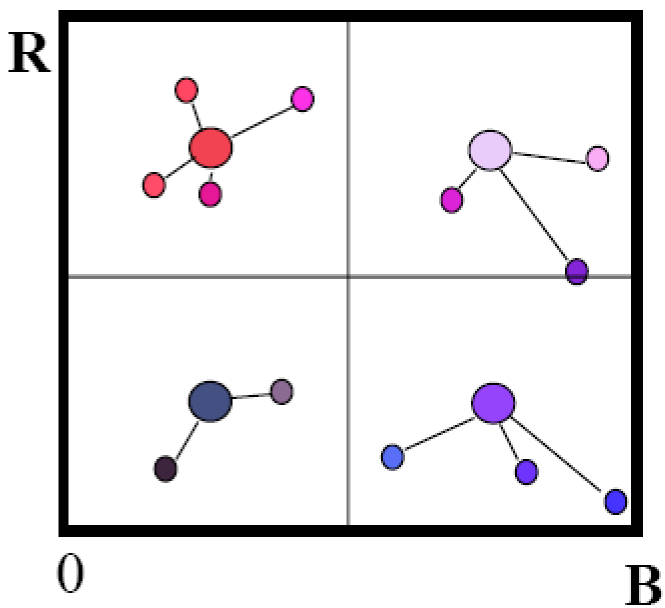
Non-uniform Quantization

- Quantization space partitioned according to input data
- Goal: choosing “best” representatives
  - Minimal distance error (if “distance” is defined)



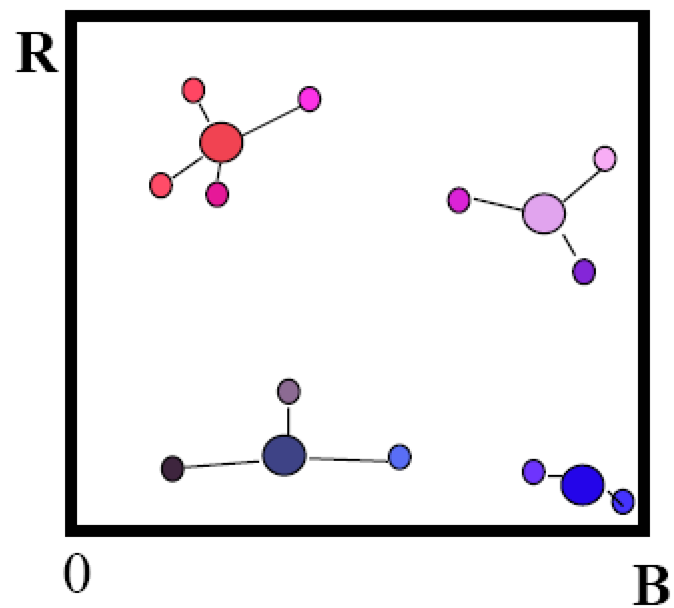
Examples

uniform quantization to 4 colors



large *quantization error*

image-dependent quantization to 4 colors



small *quantization error*

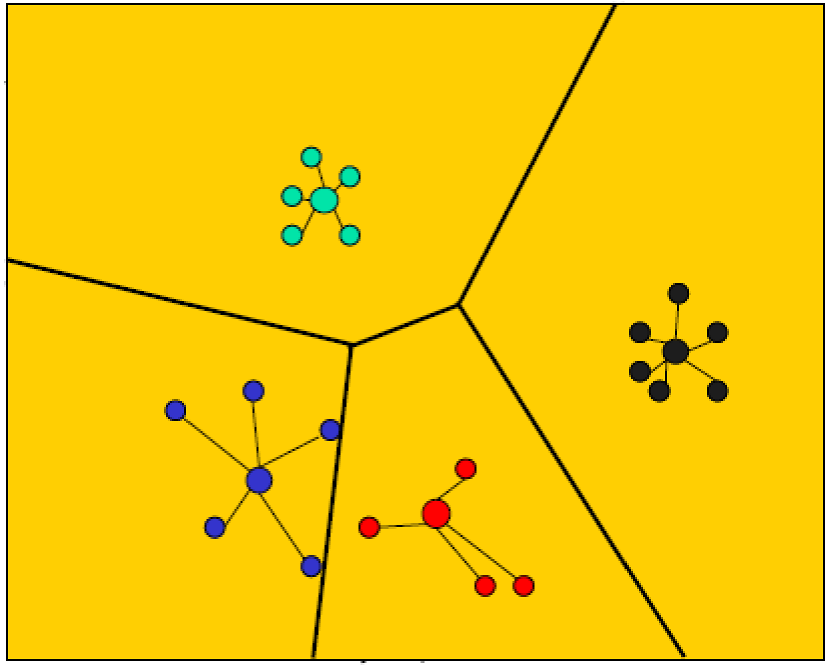
Quantization & Lossy Coding

- Quantization used as lossy coding method when there is notion of distance between symbols to be coded
  - Coordinates
  - Colors
  - Normals
  - Not good for characters

Lloyd algorithm for VQ

- Given *k*, finds best *k* representatives
- Iterative method: (v_i – representatives)
  - for i=1 to k do { $v_i \leftarrow$ random point }
  - While (v_i still moves)
    - $S_i \leftarrow$ closest data points to v_i
    - $v_i \leftarrow$ centroid of S_i (sum of S_i coordinates / $|S_i|$)

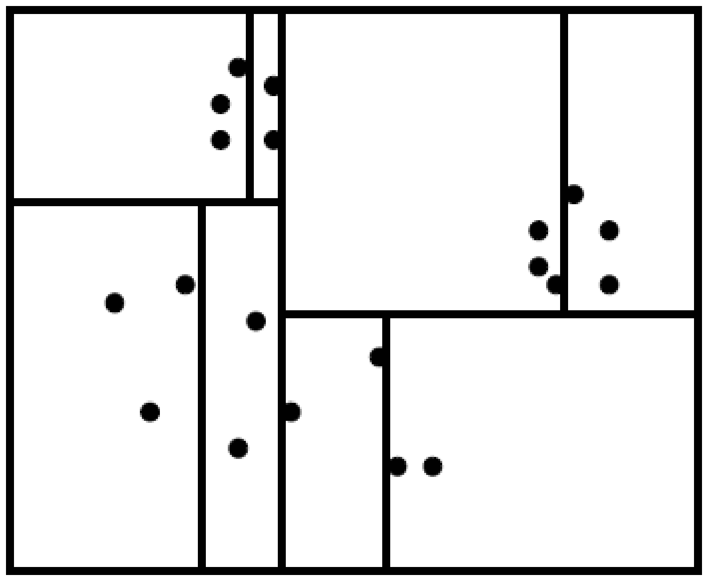
Lloyd algorithm - example



Lloyd algorithm (cont.)

- At each iteration, find S_i using Voronoi diagram (with v_i as sites)
- VQ problem in general is NP-Complete (finding BEST representatives). Lloyd algorithm generates the optimal solution but is very slow.
- What if k is not given?
  - Initialize $k \leftarrow 2$
  - Perform Lloyd algorithm
  - While quantization error is too big do:
    - $k \leftarrow k+1$
    - Perform Lloyd algorithm

Median cut quantization

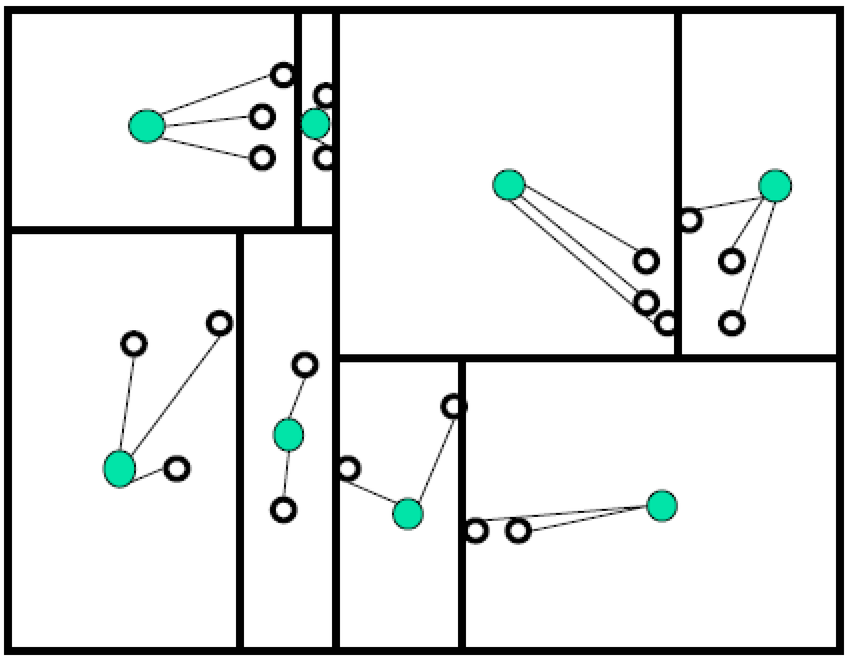


Median cut alg. - heuristic approximation to optimal (Lloyd) VQ solution

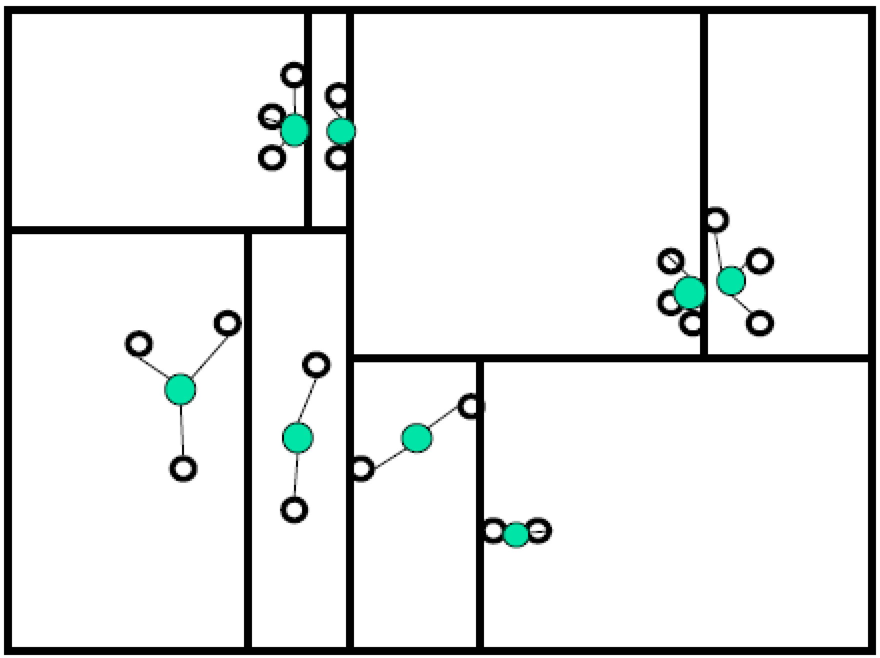
Median cut (cont.)

- *while (num_of_cells<k) do*
  - Split each cell into half vertically/horizontally alternately, according to number of sites
- Choose representatives for each cell:
  - Geometric cell center
  - Centroid of sites in cell (better results)

Median cut (cont.)



Cell center



Centroids

Uniform vs. median cut

original - 256 colors

uniform

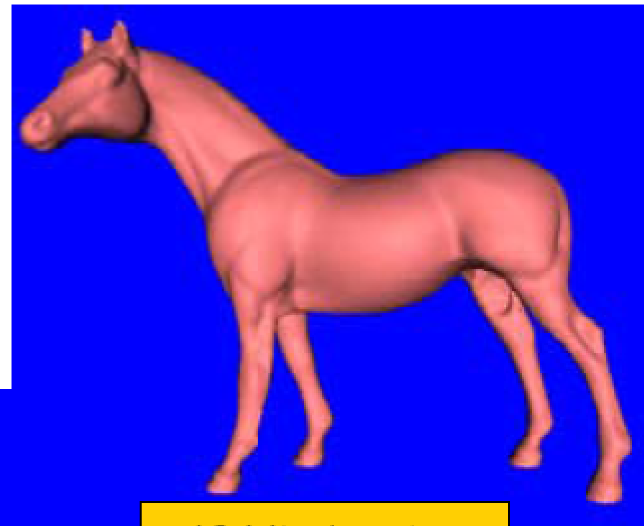
8 colors

median cut

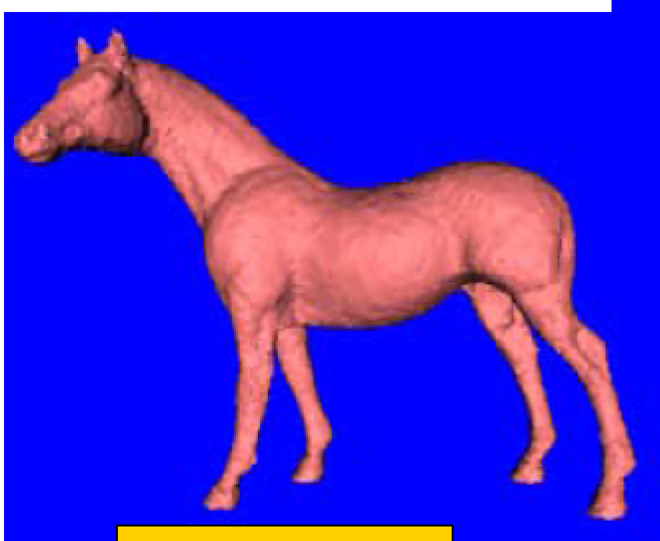
Uniform geometry quantization

- Coordinates can be considered integers in a finite range after quantization
- Quantization is done on the data bounding box/cube intervals
- Geometry quantization to n bits:
  - All integer values in $[0, 2^n-1]$ can be used
  - Scale/transform coordinates to be maximal over given range
  - Quantize each coordinate (rounding to nearest integer)

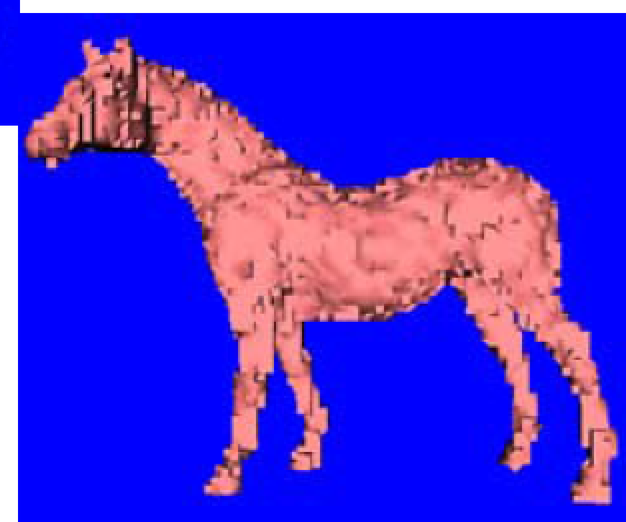
Uniform geometry quantization - results



12 bits / vertex



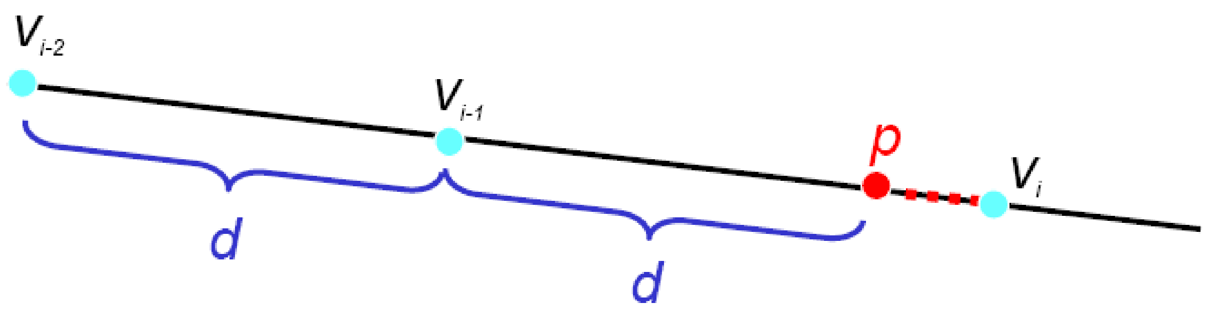
8 bits / vertex



6 bits / vertex

Prediction – History Repeats Itself

- Linear 2D predictor:



- Prediction rule: $v_i\text{-}1 - v_i\text{-}2 = p - v_i\text{-}1$
  or: $p = 2\, v_i\text{-}1 - v_i\text{-}2$
- Prediction error: $e_i = v_i - p$

Using Predicted Geometry

- $(v_1\ v_2\ v_3\ \ldots)$ - vertex coordinates
  $(e_3\ e_4\ e_5 \ldots)$ - prediction errors

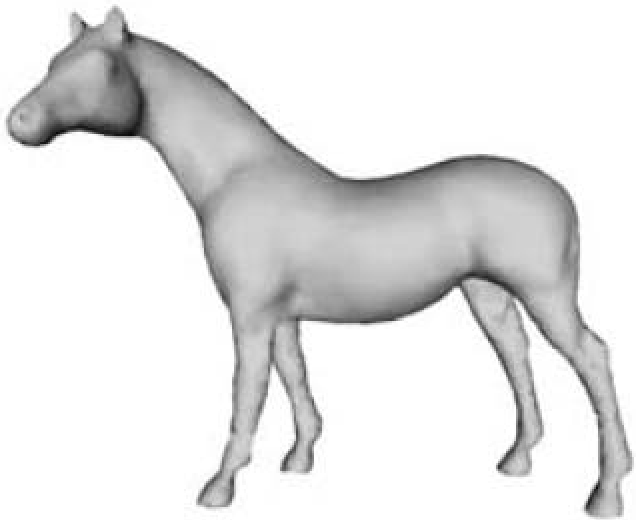
- Naive geometry coding: $v_1\ v_2\ v_3\ \ldots$

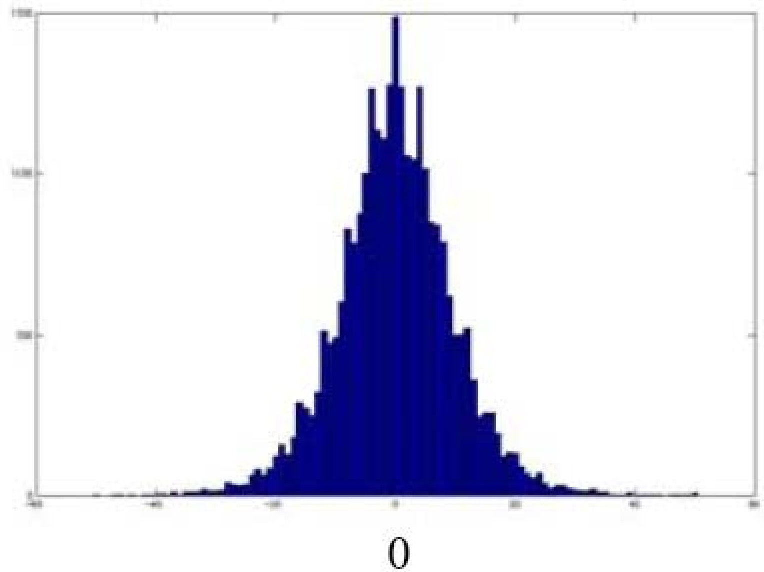
- Coding using prediction: $v_1\ v_2\ e_3\ e_4\ e_5\ \ldots$

- Decoding: $v_1\ v_2$

  $v_i = 2\ v_i\text{-}1 - v_i\text{-}2 + e_i \qquad i>2$

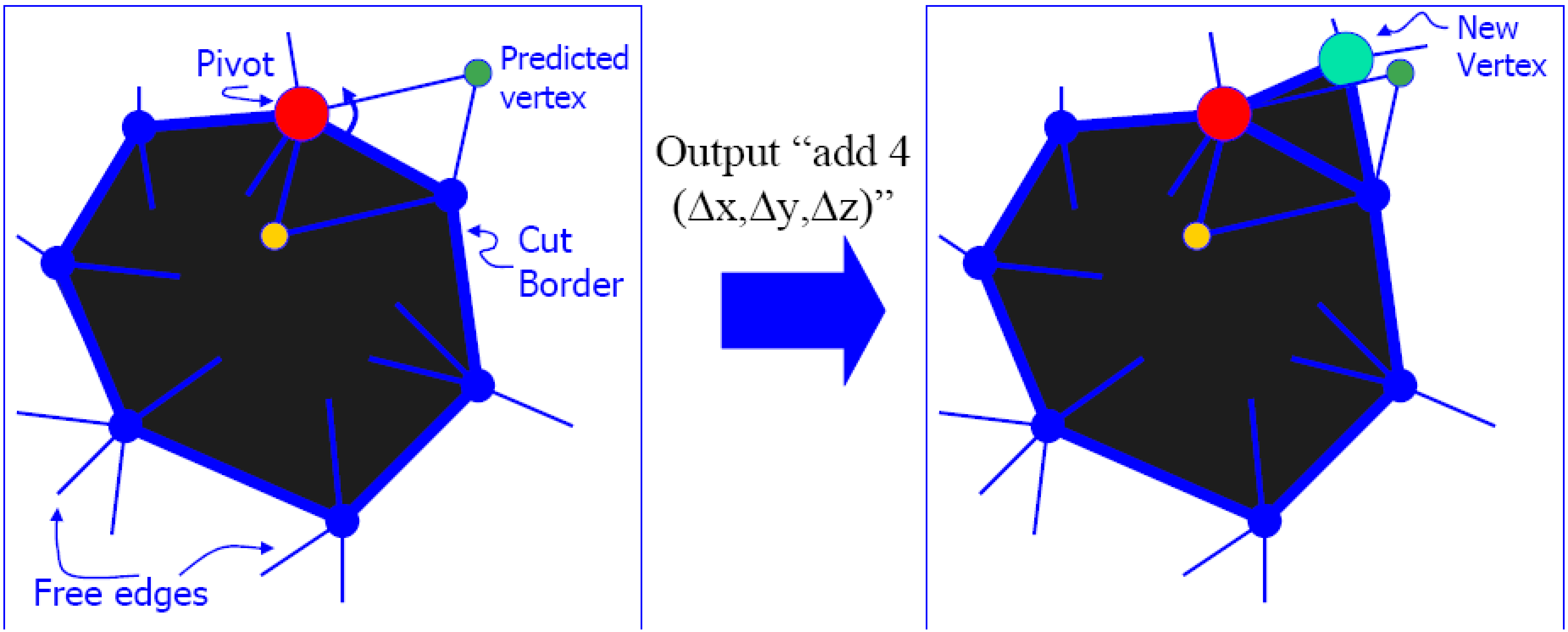
Good Prediction Reduces Entropy



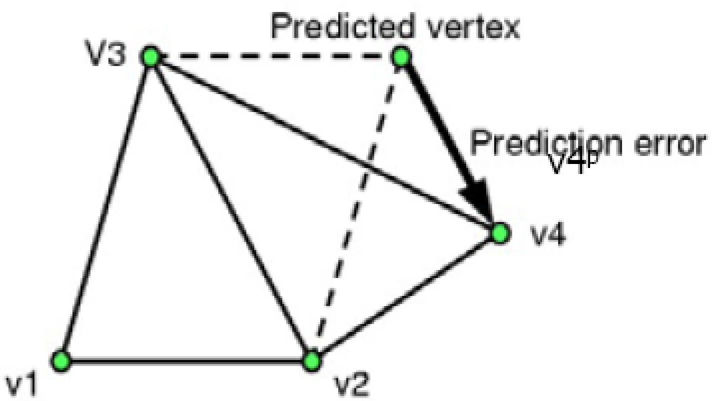


Distribution of prediction errors

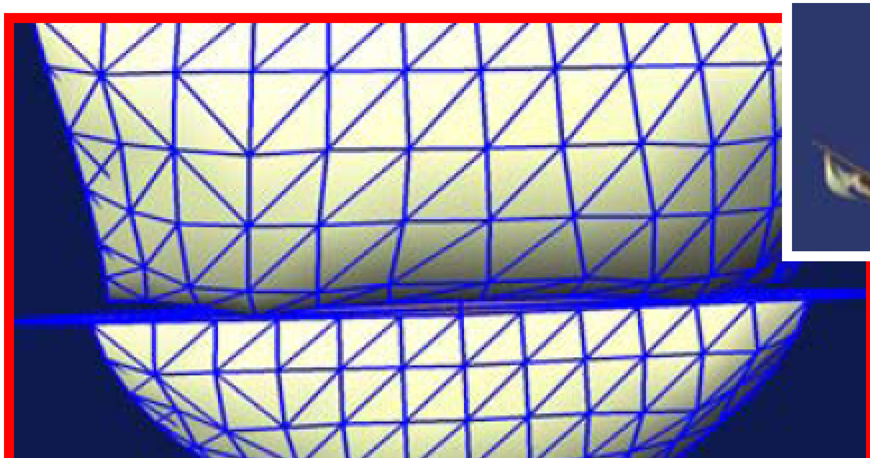
Surface-Based Prediction

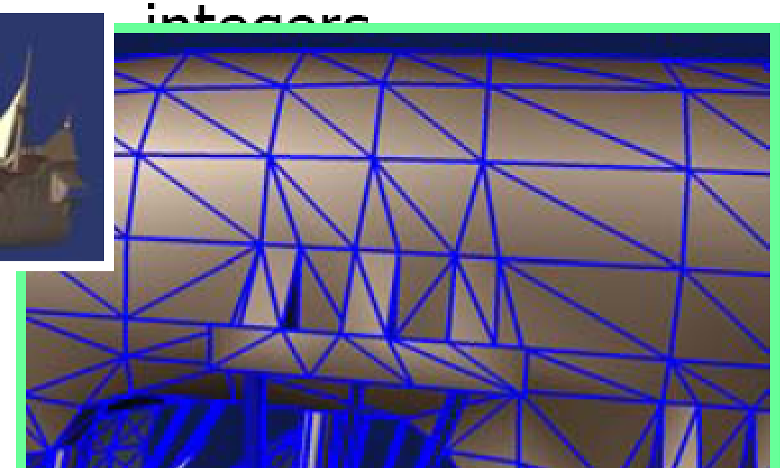


Parallelogram Prediction



- Use the connectivity to predict the geometry: $V_{4p} = v_2 + v_3 - v_1$
- $(-1, 1, 1)$ in *barycentric coords*
- **Can** be applied to integers



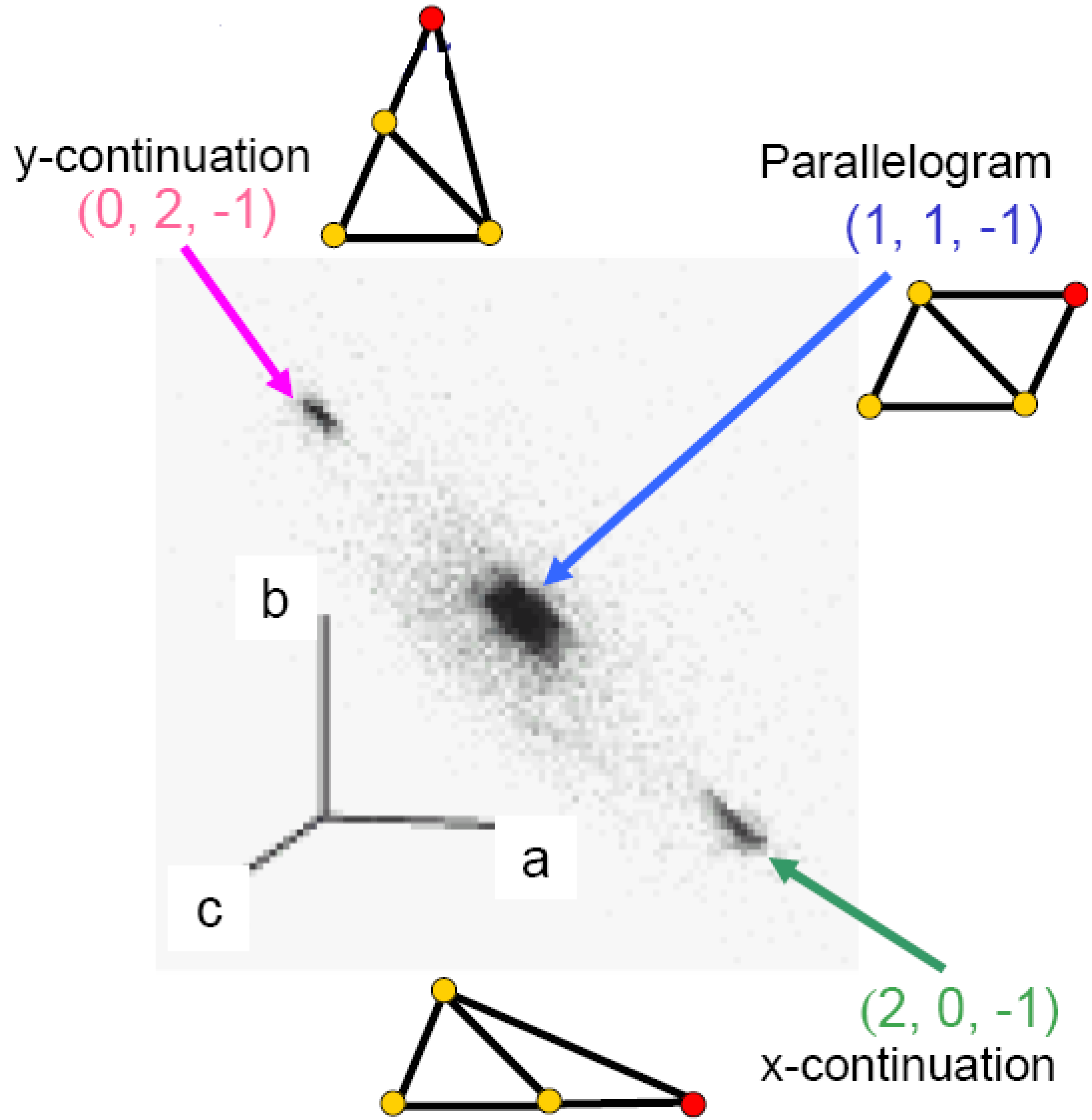


Some Results

Raw quantized data = 10 bits/coord = 30 bits/vertex

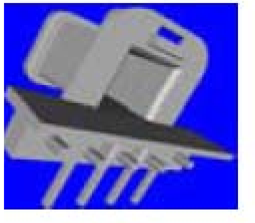
Model	vertices	line predictor	parallelogram	ratio
Eight	766	18.8	14.0	1.3
Triceratops	3100	18.4	14.1	1.3
Cow	3078	18.9	14.6	1.3
Beethoven	2847	22.7	17.3	1.3
Dodge	10466	19.8	12.4	1.6
Starship	4468	19.2	13.2	1.5
Average		19.6	14.3	1.4

Other Predictive Patterns

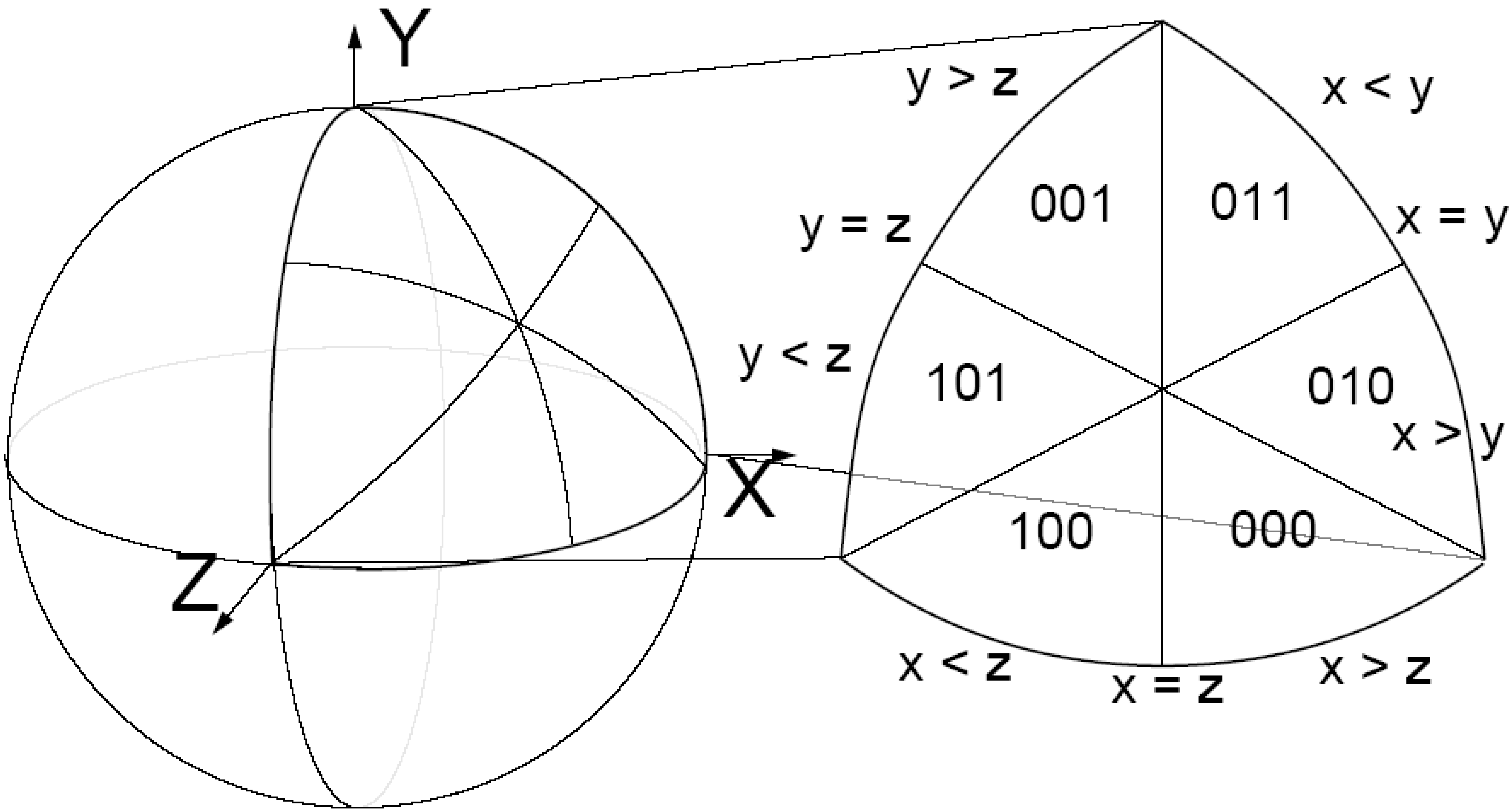


Predictor Traversal Optimization

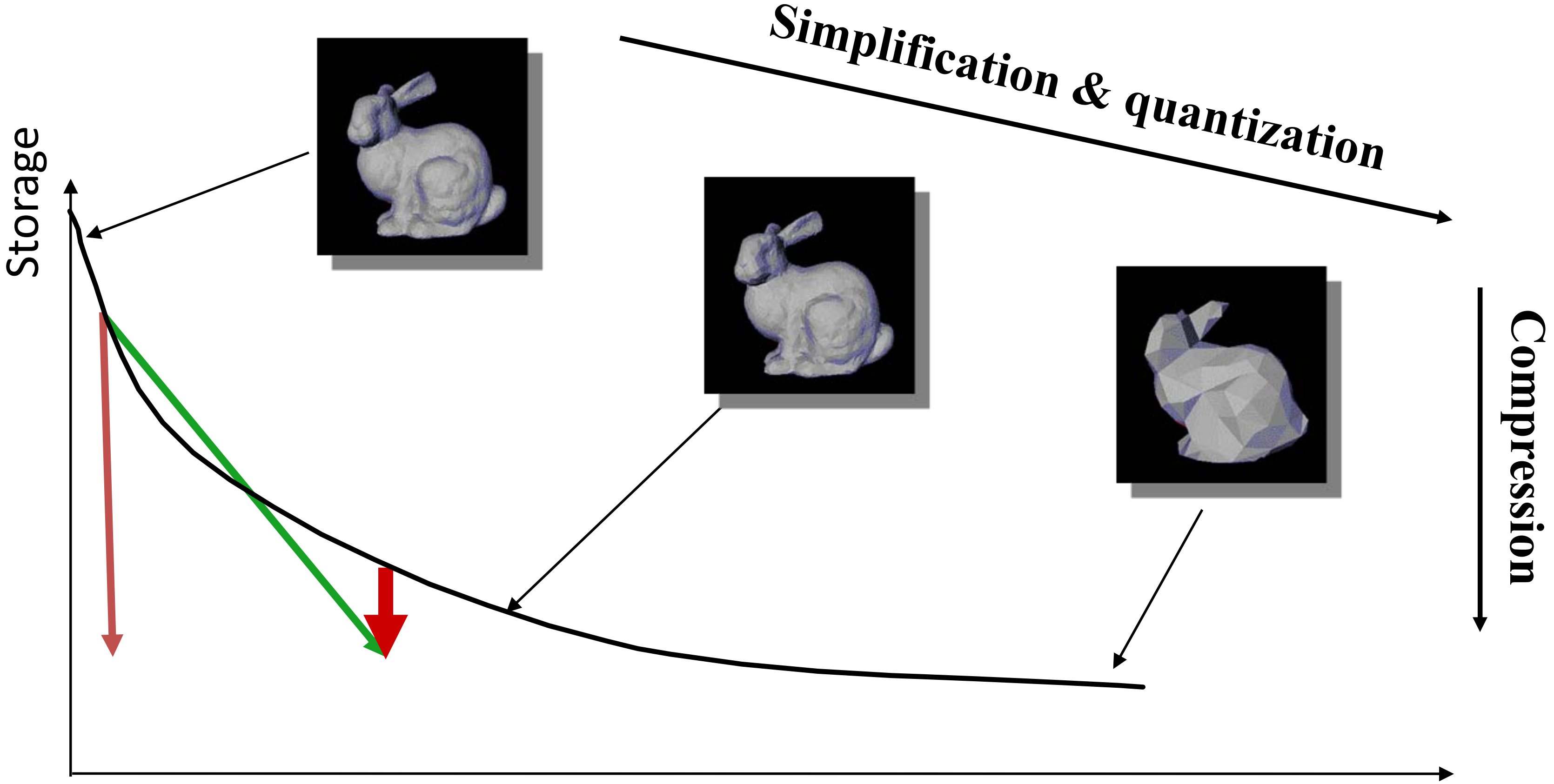
- Parallelogram predictor assumes mesh is locally planar and regular
- Problem: Fails on meshes with sharp corners and creases
- Solution: Optimize face traversal to achieve good predictors



Alternate Normal Representation



Complexity of a shape = Storage/Error curve

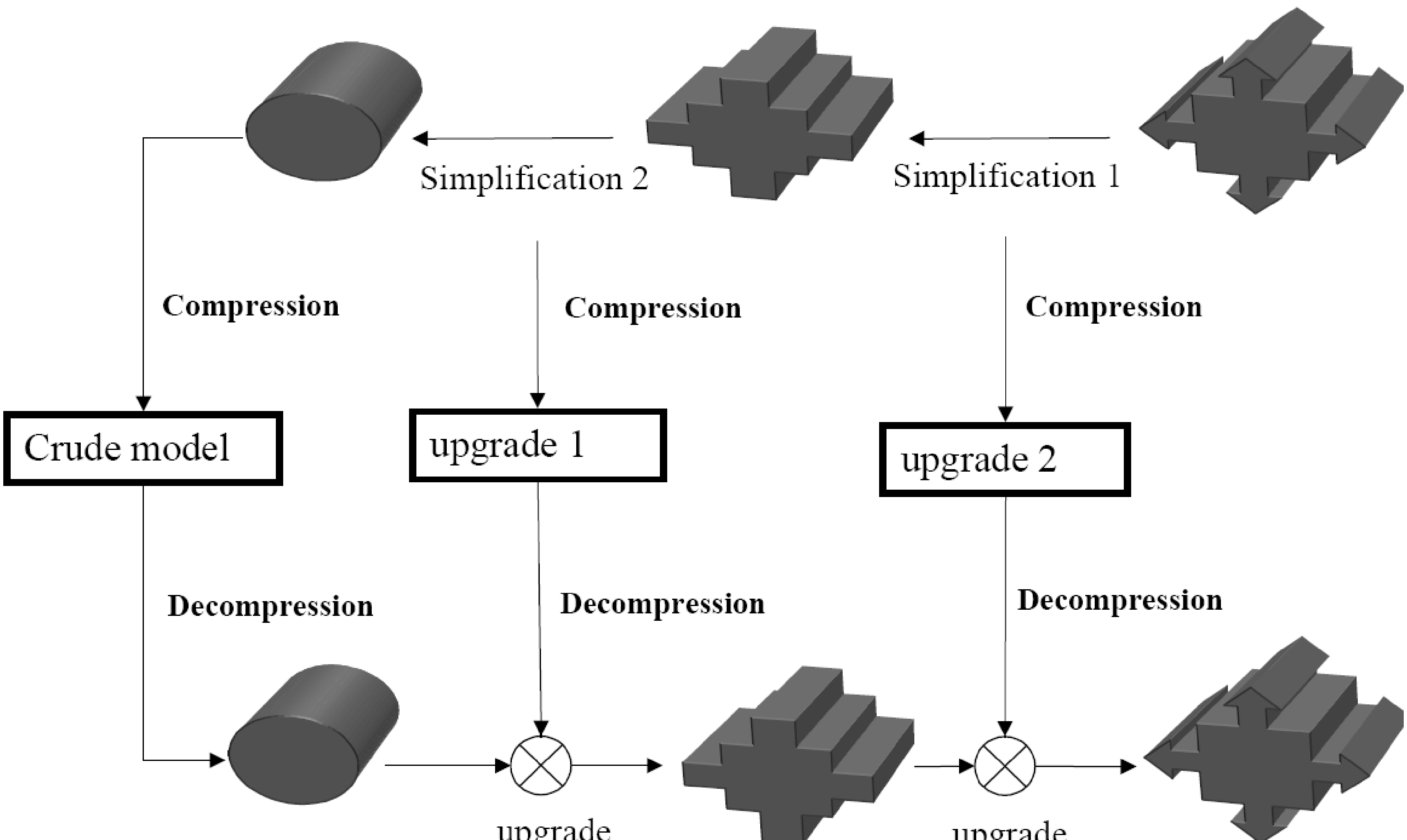


Error of the approximating model

Curve depends on representation and compression scheme used

Estimate $E_T = K/T$

Progressive Compression
- Compressed Successive Upgrades



Problems

- Higher compression ratio
- Random access
- Loss of bits

Resources

- Siggraph 2000 Course #38
- Research papers
- Internet

Discussions