A New Progressive Lossy-to-Lossless Coding Method for 2.5-D Triangle Mesh with Arbitrary Connectivity

Dan Han

University of Victoria

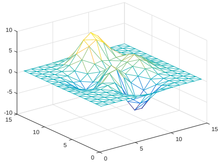
October 31, 2016

University of Victoria

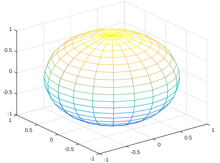
Outline

2.5-D Triangle Mesh

- one representation of bivariate functions $z = \phi(x, y)$
- compare with 3-D mesh, more restriction on data
- sample points triangulated, the domain partitioned into nonoverlapping triangle faces
- information in the mesh:
  - a set P of sample points (geometry information)
  - triangulation on the points (connectivity information)
  - a set $Z = \{z_i\}_{i=0}^{|P|-1}$ of function values where $z_i = \phi(p_i)$ (i.e., function value information)
- usually large, need to be compressed



(a) 2.5-D mesh.



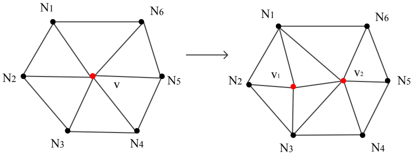
(b) 3-D mesh.

Mesh Coding

- lossy vs. lossless coding method
    - lossy: permanently discard certain information, size reduced for storage, handling, and transmission
    - lossless: decompressed mesh identical with original mesh
- progressive coding method
    - transmit complex meshes over network with limited bandwidth
    - applications requiring real-time interaction
    - meaningfully decode partial bitstream
- motivation
    - less methods are about coding 2.5-D meshes, even less for progressive coding
    - many 2.5-D mesh coding methods only handle geometry and function value information
    - our interest: progressive lossy-to-lossless coding method for 2.5-D mesh with arbitrary connectivity (PK and ADIT methods)

Vertex Split

- an operation on triangulation
- increase the number of vertices by one



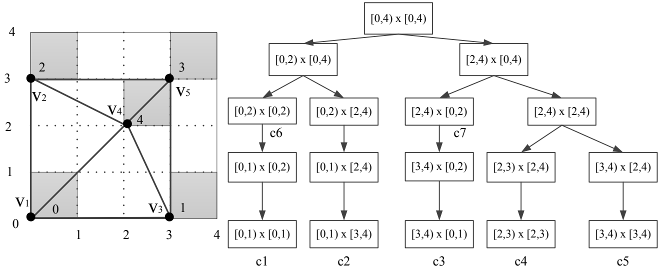
- connectivity changes
  - updates on the original neighbors, pivots and nonpivots
  - connectivity of new vertices

Proposed Mesh-Coding Framework

Main idea

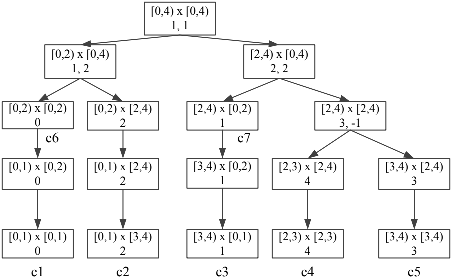
- focus on encoder procedure
- use data structure: cell bi-partitioning tree (**cbp-tree**)
- store the mesh information in the tree nodes
  - geometry
  - connectivity
  - function value
- code information in the tree nodes using a top-down traversal

Mesh and Cbp-Tree I



- recursive cell bi-partitioning
- each node corresponds to one nonempty cell $[x_1, x_2) \times [y_1, y_2)$
- leaf node: unit cell, one-on-one correspondence with the original sample point
- store function values and connectivity into leaf nodes
- propagate information stored in leaf nodes upwards

Mesh and Cbp-Tree II



- each node:
  - a cell
  - a representation vertex
  - one approximation coefficient
  - zero or one detail coefficient
- leaf node
- nonleaf node
  - has two child nodes
  - has one child node
- connectivity of nonleaf nodes

Another Perspective of Cbp-Tree

- collapse two levels of cbp-tree cell-partitioning (CCP) into one level
- quadtree cell partitioning (QCP)
- first along x, then along y-axis

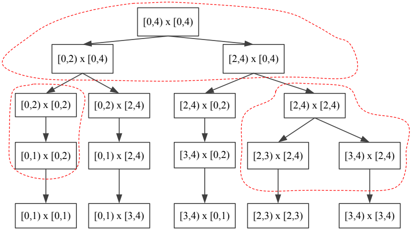


Figure: Previous cbp-tree with the dashed lines grouping the CCPs into QCPs

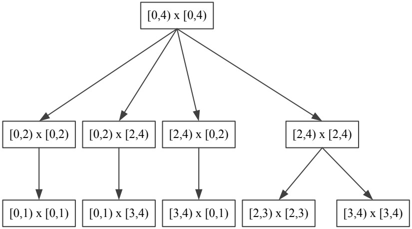


Figure: quadtree-view with the QCP operations

Encoding Geometry Information

- quad-tree view, richer geometry information
- M maximum nonempty cells
- T actual nonempty cells
- information to be coded
  - number T of nonempty child cells
  - which T of M are nonempty

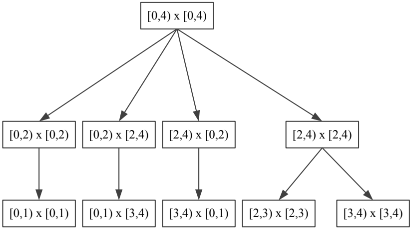
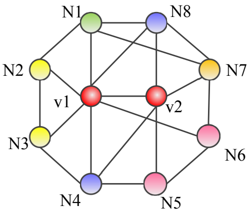


Figure: quadtree-view

Encoding Connectivity Information

- cbp-tree view
- if a node has two child nodes, vertex split
- connectivity changes when a vertex v is split into two new vertices v_1 and v_2
- v has M neighbors: $\{N_1, N_2, \ldots, N_M\}$.
- information to be coded
  - number P of pivots, and which P of M neighbors are pivots
  - how the nonpivots connect to v_1 or v_2
  - how v_1 and v_2 connect to each other



Encoding Function Value Information

Information to be coded

- only code approximation coefficient for root node
- for each node, code detail coefficient if there is any

- function value, n-bit unsigned integer
- detail coefficient, $(n+1)$-bit signed integer
- code starting from most significant magnitude bit
- sign bit after the first nonzero magnitude bit

Sign bit

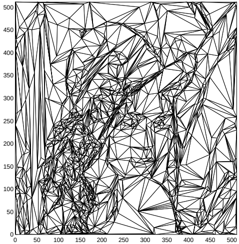
0/1 $b_{n-1} \ldots b_2 b_1 b_0$

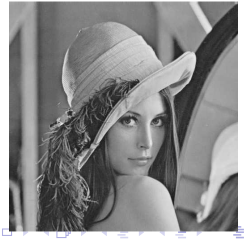
Decoding Procedure

- almost a mirror of the encoding
- start from one single root node
- build the tree progressively based on the received information
- when the decoding is terminated:
  - current leaf nodes are represented as vertices located at the centroids of the cells
  - edges are generated based on the connectivity relation between the leaf nodes

Test Data

- most interested in using the meshes to model images
- 64 mesh models of images studied
- measure mesh quality by measuring reconstructed image using PSNR
- meshes grouped into three categories A, B, and C based on the connectivity information
  - A: Delaunay connectivity
  - B: non-Delaunay connectivity with good quality
  - C: non-Delaunay connectivity with bad quality





Proposed Method vs. Gzip

- Gzip is a general-purpose compression method

Table: Overall average results for all meshes in three categories

Category	Original Mean Rate (bits/vertex)	Mean Rate (bits/vertex)		Gzip/Proposed Ratio
		Gzip	Proposed	
A	349.64	119.52	**19.07**	6.27
B	351.77	144.32	**23.47**	6.15
C	348.52	136.27	**31.82**	4.28
Overall	350.30	140.30	**23.72**	5.92

- the proposed method: 23.72 bits per vertex (bpv)
- Gzip method: 140.30 bpv
- the proposed method is roughly **5.92** times better than Gzip.
- Gzip cannot perform progressive coding

Proposed Method vs. Edgebreaker-Based Method I

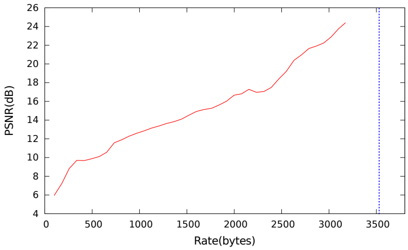
- Edgebreaker is a 3-D mesh coding method
- lossless coding performance is related to mesh connectivity

Edge-flipping Distance Range	Mean Rate (bits/vertex)	
	Edgebreaker	Proposed
0	20.51	**19.05**
$(0, 37.38\%]$	21.88	**20.87**
$(37.38\%, 80\%]$	**22.67**	23.29
$> 80\%$	**28.96**	31.83

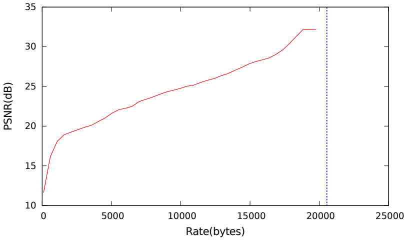
- edge-flipping distance: percentage of edges to be flipped
- average lossless coding rate
  - Delaunay: proposed is better, 7.7% less bit rate
  - $(0, 37.38\%]$: proposed is better, 4.8% less bit rate
  - $(37.38\%, 80\%]$: Edgebreaker is better, 2.7% less bit rate
  - $> 80\%$: Edgebreaker is better, 9.9% less bit rate

Proposed Method vs. Edgebreaker-Based Method II

- Edgebreaker cannot progressive coding



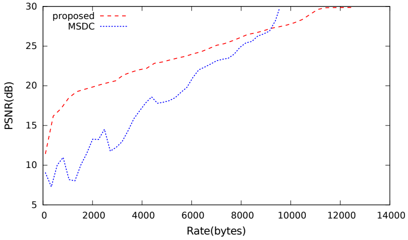
(a) Mesh 1 (46.26%)



(b) Mesh 2 (30.80%)

Proposed Method vs. MSDC Method I

- the MSDC method:
  - 2.5-D mesh coding method
  - only codes geometry information and function value information
  - assumes Delaunay connectivity
  - has lower lossless coding bit rate



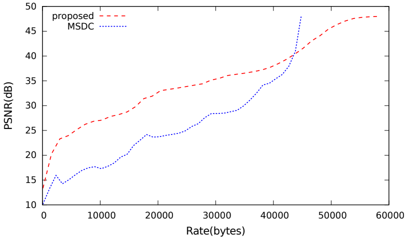


Figure: Comparison of progressive coding performance

Proposed Method vs. MSDC Method II

- low bit rate:

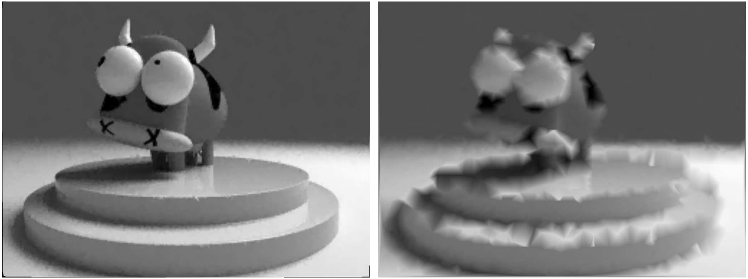


Figure: 29.34 dB vs. 23.07 dB

Proposed Method vs. MSDC Method III

- high bit rate:

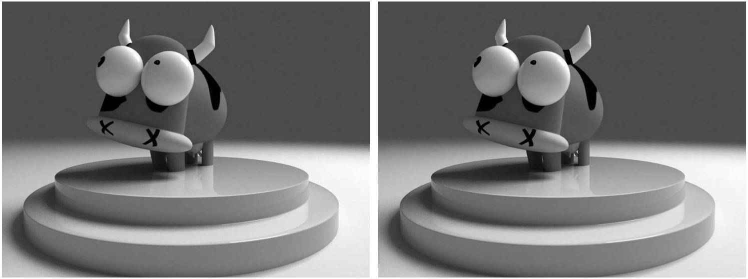


Figure: 41.08 dB vs. 47.99 dB

- overall: 75% maximum PSNR, proposed method uses less bit rate, 55% to 86% of the bit rate used by the MSDC

Conclusion

- proposed new progressive lossy-to-lossless coding framework for 2.5-D triangle meshes with arbitrary connectivity
- performance of the proposed method:
  - outperforms Gzip, 5.92 times better on average
  - outperforms the Edgebreaker-base method, when meshes not deviated too far from Delaunay triangulation (37.38%)
  - Gzip and Edgebreaker cannot progressive coding
- outperforms the MSDC method for progressive coding performance, using 55% to 86% bit rate to achieve 75% maximum PSNR

THANK YOU!

Q&A

References:

[1] M. D. Adams, *An efficient progressive coding method for arbitrarily-sampled image data*, IEEE Signal Processing Letters, Vol. 15, pp. 629632, 2008.

[2] M. D. Adams, *Progressive lossy-to-lossless coding of arbitrarily-sampled image data using the modified scattered data coding method*, IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 10171020, 2009.

[3] M. D. Adams, *An improved progressive lossy-to-lossless coding method for arbitrarily-sampled image data*, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 7983, 2013.

[4] J. Peng and C.-C. J. Kuo, *Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition*, In ACM Transactions on Graphics, Vol: 24, pp. 609616, 2005.

[5] L. Demaret and A. Iske, *Scattered data coding in digital image compression*, Curve and Surface Fitting: Saint-Malo, Vol:2003, pp: 107-117, 2002.

[6] Y. Tang, *Edgebreaker-based triangle mesh-coding method*, 2016.

Triangulation I

A **triangulation** of a finite set P of points in $\mathbb{R}^2$ is a set T of (non-degenerate) open triangles such that:

1. the set of all the vertices of triangles in T is P;
2. the interiors of any two triangles in T are disjoint;
3. the union of all triangles in T is the convex hull of P; and
4. every edge of a triangle in T only contains two points from P.

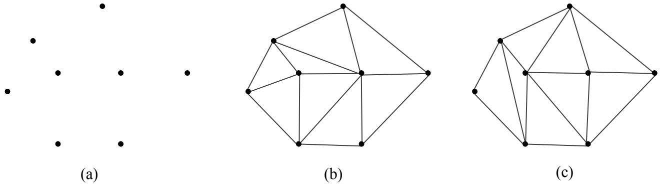
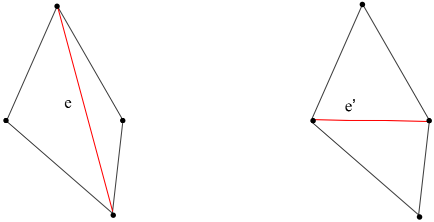


Figure: Triangulation examples. (a) A set P of points, (b) a triangulation of P, and (c) another triangulation of P.

Triangulation II

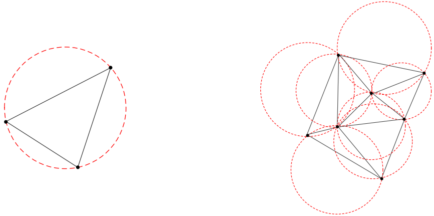
- one basic operation on a triangulation is an **edge flip**
- edge e is called flippable if e has two incident faces and the union of these two faces is a strictly convex quadrilateral Q.



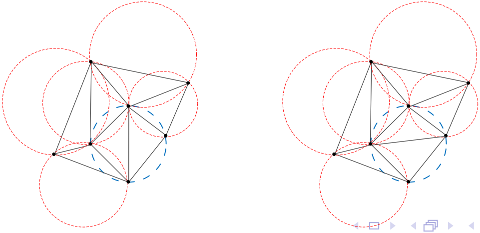
- for the same set P of points, one triangulation T can always be transformed into another triangulation T' with a finite sequence of edge flips

Delaunay Triangulation

- no point in P is strictly inside the circumcircle of any triangle in T.

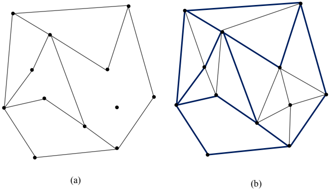


- not guaranteed to be unique, only guaranteed to be unique if no four points in P are co-circular.



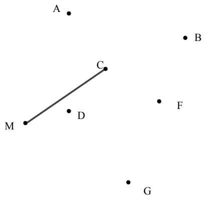
Constrained Delaunay Triangulation I

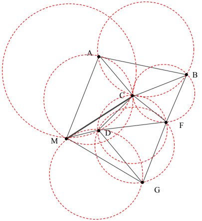
- a triangulation with constrained edges is called a **constrained triangulation**
- planar straight line graph(PSLG):
    - a PSLG (P, E) is a collection of a set P of points in $\mathbb{R}^2$ and a set E of line segments such that:
        1. the endpoints of each segment of E must be in P; and
        2. any two segments of E must be disjoint or intersect at most at a common endpoint



Constrained Delaunay Triangulation II

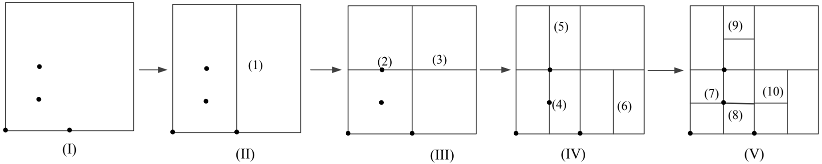
- visibility: two points A and B are visible to each other in the PSLG (P, E), if and only if segment $\overline{AB}$ does not intersect the interior of any constrained edges in E
- constrained Delaunay triangulation:
  - given a PSLG (P,E), a triangulation T of P is said to be **constrained Delaunay** if each triangle t in T is such that: 1) the interior of t does not intersect any constrained edges in E; and 2) no vertex inside the circumcircle of t is visible from the interior of t.



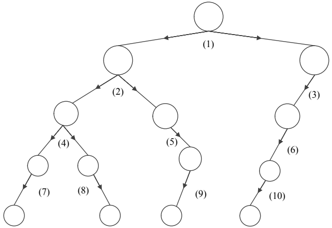


Go back

Cbp-Tree



- based on spatial partitioning
- initial cell: root cell
  - contains all the sample points
- recursively bi-partitioning along x and y-axes
  - empty cell, no sample points
  - degenerate cell, zero area
- until nonempty cell contains only one sample point

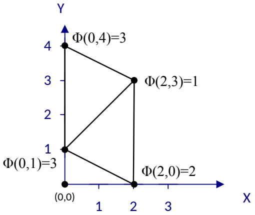


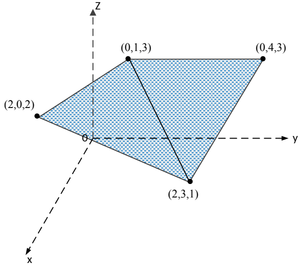
- root node
- each node, nonempty cell

Go back

2.5-D Mesh Model I

- 2.5-D meshes can be used to represent many types of data
- most interested in using the meshes to model images
- measure the difference between two meshes by comparing vertices is tricky to do
- instead measure the difference between the image reconstructions produced by the meshes
- generate a function $\tilde{\phi}$ defined on the entire domain D (and not just at lattice points in S), the function values Z are used in conjunction with linear interpolation.





2.5-D Mesh Model II

- function values in the original image are integers
- image approximation function $\hat{\phi}$ also need to be integer-valued
- rounding the non-integer function values of $\tilde{\phi}$ to the nearest integers: $\hat{\phi} = \text{round}\left(\tilde{\phi}\right)$.
- standard rasterization techniques
- mean squared error(MSE)
  - $\text{MSE} = |P|^{-1} \sum_{p \subset P} \left(\phi(p) - \hat{\phi}(p)\right)^2$
- peak signal-to-noise ratio (PSNR)
  - $\text{PSNR} = 20 \log_{10} \frac{(2^{\rho} - 1)}{\sqrt{\text{MSE}}}$
- smaller MSE, higher PSNR, better quality of reconstructed image

Go back

Test Data Example

	Mesh Information			Statistics about Valence						
Nickname	#Vertices	#Edges	#Faces	Max.	Min.	Median	Mean	Width×Height	ρ	Distance †
B3	15728	47042	31315	13	2	6	5.98194	1024×768	8	0
L1	1310	3880	2571	11	2	6	5.92366	512×512	8	0
P4	10485	31322	20838	11	3	6	5.97463	512×512	8	0
A2	14794	44311	29518	27	3	6	5.9904	1238×1195	8	39.55%
B5	3932	11757	7826	21	3	6	5.98016	1024×768	8	49.86%
CR2	35717	106893	71177	22	3	6	5.98555	1744×2048	10	33.73%
CT3	5242	15703	10462	29	2	6	5.99122	512×512	12	40.01%
Q4	38400	115193	76794	141	3	6	5.99964	1200×1600	8	41.53%
L9	1310	3919	2610	395	3	5	5.98321	512×512	8	145.65%
M7	14550	43580	29031	638	3	4	5.99038	1912×761	8	155.22%

- nickname of the mesh for convenience
- number of vertices, edges, and faces in the mesh
- statistics of valences in the original mesh: maximum, minimum, average, and median value of valences
- bounding box of conv(P): width, height
- number of bits to represent function values
- edge-flipping distance between the current mesh connectivity and (preferred-direction) Delaunay triangulation

Go back

Vertex Split Cont'd

Connectivity Change

- Updates on the original 1-ring neighbors $\{N_i\}$:
  - connect to v_1 or v_2 (nonpivot); or
  - connect to both v_1 and v_2. (pivot)
- Connectivity of two new vertices: v_1 and v_2 connected or not.
- Sometimes cause invalidate triangulation

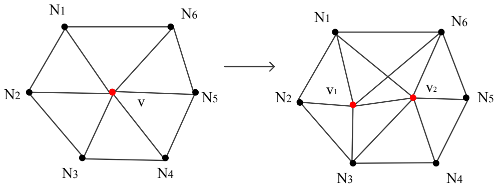
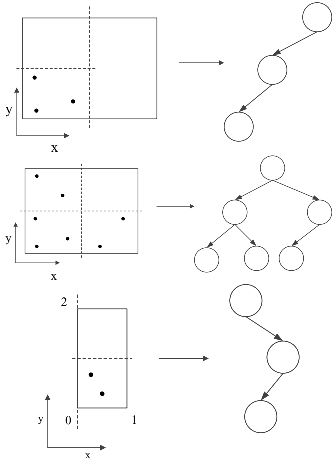


Figure: An example of vertex split leading to invalid triangulation.

Go back

Encoding Procedure Overview

- top-down traversal
- geometry information
  - consider QCP operation, provide richer information
  - how many nonempty cells T
  - which of them are nonempty
- connectivity information
  - $\max(T-1, 0)$ vertex splits
  - how original neighbors connect to v_1 and v_2
  - how v_1 and v_2 connect to each other
- function value information
  - approximation coefficient of root node a_r
  - detail coefficient of each non-root node, if any



binarization of a hexary symbol

- has six possible values, e.g., $\{0, 1, \ldots, 5\}$
- binarization scheme for a hexary symbol:
  - code $b_1 = \lfloor n/3 \rfloor$ in bypass mode
  - code a ternary symbol $b_2 = \text{mod}\,(n, 3)$.
- binarization scheme for a ternary symbol:
  - code $b_1 = \lfloor n/2 \rfloor$, using the context c_{third}
  - if $b_1 = 0$, code another symbol $b_2 = \text{mod}\,(n, 2)$ in bypass mode

Go back

- n-bit unsigned integer, $v = \sum_{i=0}^{n-1} b_i 2^i$
- **UI-binarization** scheme UI(n, f), proposed in IT-method
- single parameter f, $f \in [1, n]$
- binarization steps:
  1. code the kth bit b_k using context c, where

$$c = \begin{cases} 2^{f-1} - 1 + \sum_{i \in [k+1, f)} (2b_i - 1) 2^{i-1}, & k \in [0, f-1), \\ 2^f - f + k - 1, & k \in [f, n-1]. \end{cases}$$

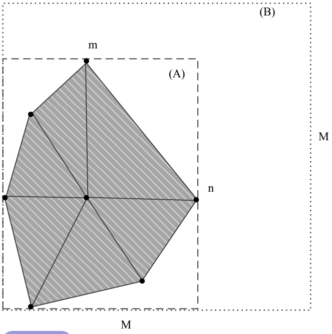
  2. if $b_k = 1$ and $k \geq f$, the remaining bits $\{b_i\}_{i \in [0,k)}$ will be coded in bypass mode and the loop will be terminated earlier
- each symbol in the range $\left[0, 2^f\right)$ are assigned a distinct probability
- others (if any) are partitioned into the ranges $\left[2^i, 2^{i+1}\right)$ for $i \in [f, n)$, equal probable
- if $f = n$, each symbol is coded using a distinct probability
- **SI**(n, f)**-binarization** works similar, extra sign bit coded in bypass mode

Go back

root cell

Two schemes are provided for selecting the root cell:

1. Unpadded scheme: smallest isorectangle containing conv (P)
2. Padded scheme: smallest isorectangle containing conv (P) that also has dimensions that are equal and integer powers of two.



- gray area is conv (P)
- the dashed line (A) represents the unpadded root cell with the size $m \times n$
- line (B) represents the padded root cell with the size $M \times M$ and contains (A)
- M is the smallest integer power-of-two that is no smaller than m or n

Go back

Mesh and Cbp-tree Cont'd

- leaf nodes: have one-on-one correspondence with the original sample points
- unit cell, function values, connectivity
- propagate upwards

Store Mesh Info into Tree Nodes

- geometry information
  - location of sample points
- function values
  - approximation Coefficients
  - detail Coefficients
  - AD transform
- connectivity information.

Average-Difference(AD) transform

$$y_0 = \left\lfloor \frac{1}{2}(x_0 + x_1) \right\rfloor$$
$$y_1 = x_1 - x_0$$

reverse operation

$$x_0 = y_0 - \left\lfloor \frac{1}{2} y_1 \right\rfloor$$
$$x_1 = y_1 + x_0$$

Generate the cbp-tree

- each node in the tree associates with a nonempty cell
- root cell $\rightarrow$ root node

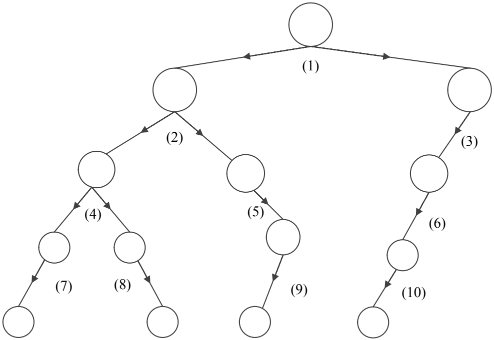
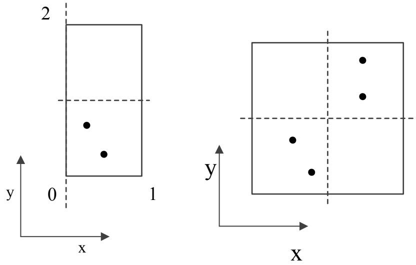


Figure: A cbp-tree structure

quadtree cell partitioning

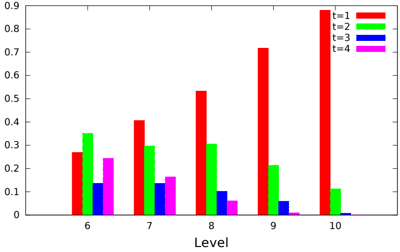
- maximum number M of nonempty cells, $M = 2$ or $M = 4$



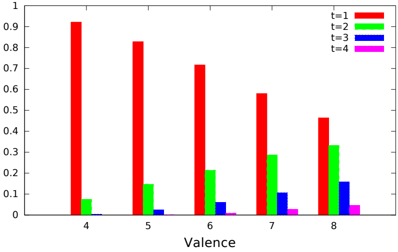
Go back

encoding of T value

- $T \in \{1, 2, \ldots, M\}$, M is the maximum number of nonempty cells



(a) Same valence (i.e., 6) with different levels.



(b) Same level (i.e., 9) with different valences.

Figure: Distributions of T under different valences and levels.

- arithmetic coding conditioned on level (QP-level) and valence
- decrease of coding bit rate by 61.9%

Go back

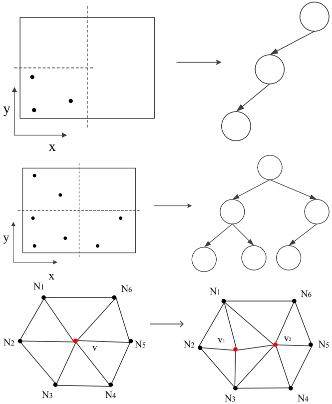
encoding of T-tuple

- $\binom{M}{T}$ possible configurations
- if $M = 2$
  - if $T = 1$, two possible configurations, 1 binary symbol
  - if $T = 2$, one possible configuration
- if $M = 4$
  - if $T = 1$ or $T = 3$, 4 possible configurations, two binary symbols
  - if $T = 2$, six possible configurations, one hexary symbol

    binarization of hexary symbol
  - if $T = 4$, one possible configuration

Encoding Connectivity Information

- assume T nonempty cells after QCP
- $\max\{T-1, 0\}$ vertex splits
- connectivity changes when a vertex v is split into two new vertices v_1 and v_2
- v has 1-ring neighbors: $\{N_1, N_2, \ldots, N_M\}$.
- information need to code
  - number P pivots, and which P of M neighbors are pivots
  - how the nonpivots connect to v_1 or v_2
  - how v_1 and v_2 connect to each other (one bit)



Go back

encoding of P value

- P related to M
- distribution of P
- coding bit rate decreased by 74.3% on average

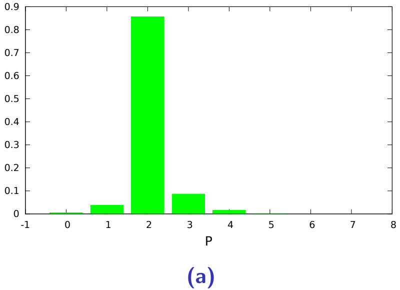


Figure: Distributions of P when (a) $M = 7$ and (b) $M = 8$.

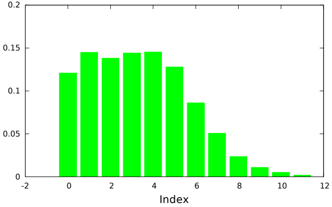
Go back

encoding of P-tuple I

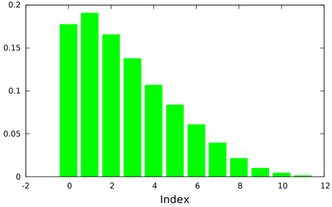
- total number of different configurations is $\binom{M}{P}$
- each P-tuple is assigned an index i, so $i \in \left\{0, 1, 2, \ldots, \binom{M}{P} - 1\right\}$
- encode the index of pivot-vertex-tuple

For each neighbor N_i, the possibility to be a pivot is determined by the formula

$$p_i = r_i = \frac{\sigma_i}{2s} = \frac{\sqrt{s(s-a)(s-b)(s-c)}}{2s},$$



(a) without priority



(b) with priority

encoding of P-tuple II

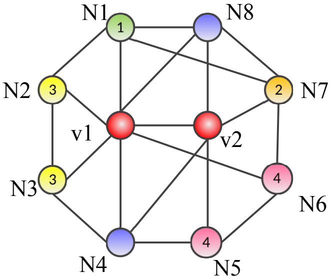
- with priority scheme, decreased by 53.10% compared with bypass, decreased by 2.0% compared with no-priority

Go back

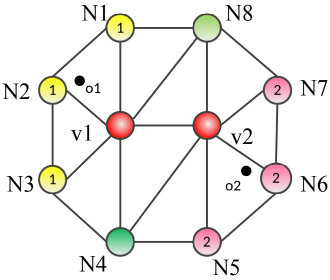
encoding nonpivots I

partitioning rules

- partition first into different segments
  - each nonpivot connects to more than two other vertices in N_i forms a segment by itself; and
  - other nonpivots are partitioned into maximum-connected segments.

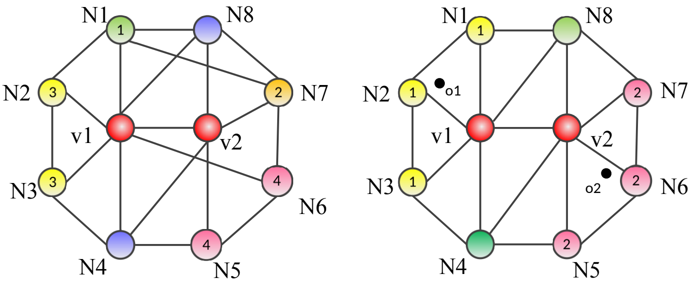


(a) four segments



(b) two segments with centroids

encoding nonpivots II



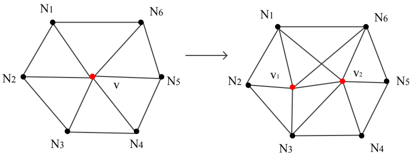
For each segmentation:

- one bit indicate whether connect to the same one of v_1 or v_2. If not, treat as separate segments. (around 0.03 bpv)
- estimate which of the segment connects to, one bit to indicate whether the estimation is accurate or not (around 2.38 bpv, decreased by 44% compared with bypass)

Go back

potential problem in decoding

- invalid triangulation connectivity caused by vertex split



Solution

- use constrained Delaunay triangulation
- previous-inserted edge removed if new-inserted edge intersects with it.

Go back

Time Complexity

- hardware used: 13-year-old computer with a 3.16 GHz Intel Core2 Duo CPU and 4.0 GB of RAM
- average time used for meshes in different categories:
  - category A: 0.75 s
  - category B: 1.33 s
  - category C: 1.90 s
- compared with Edgebreaker: 0.16 s, 0.25 s, and 0.19s.
- more computational complex because of the progressive functionality

Go back