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

Xiao Feng and Michael D. Adams

University of Victoria

xiaofeng@uvic.ca and mdadams@ece.uvic.ca

August 24, 2015

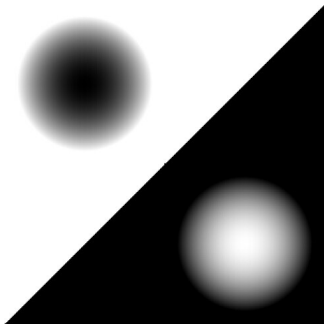
Outline

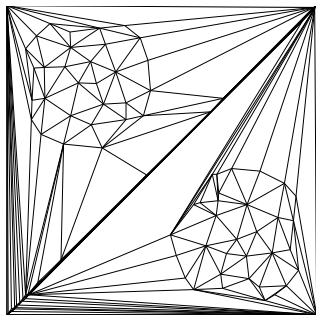
Introduction

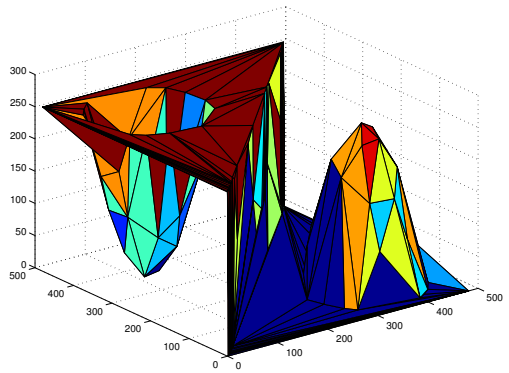
- Growing interest in image representations based on triangle meshes
- Such representation known as **mesh models**
- We want to encode mesh models of images
- 2.5-D mesh, not arbitrary 3-D mesh
- Effective approach **IT method**, however, has no means for coding mesh connectivity
- In our work, effective techniques for coding mesh connectivity are developed

Mesh Models of Image

- Image function ϕ defined on domain $\Gamma = [0, W-1] \times [0, H-1]$ and sampled on rectangular grid Λ of width W and height H
- Mesh model of ϕ is characterized by
  1. set P of sample points ($P \in \Lambda$)
  2. triangulation of P
  3. set Z of function values for ϕ at each point in P (i.e., $z_i = \phi(p_i)$)

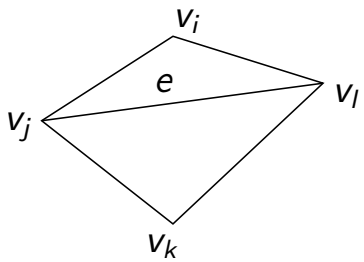




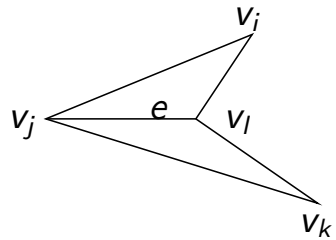


- A continuous piecewise-linear function $\hat{\phi}$ is constructed to approximate ϕ

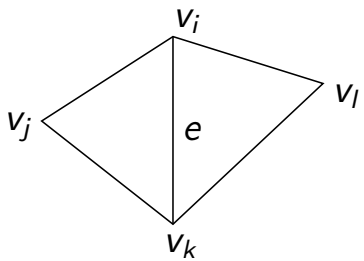
Edge Flip



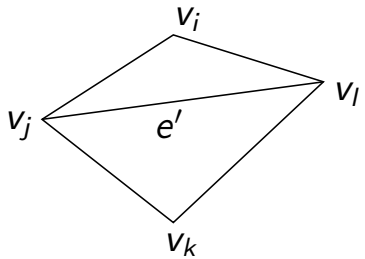
flippable edge



non-flippable edge



before edge flip

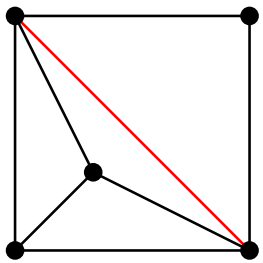


after edge flip (replace e with e')

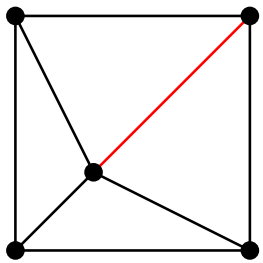
- Given two triangulations of a set of points, always possible to transform one triangulation to another by finite sequence of edge flips

Local Optimization Procedure (LOP)

- A technique to find optimal triangulation of a set of points
- Optimality criterion for edges must be defined with LOP
- LOP keeps applying edge flips to flippable edges that are not optimal until all flippable edges in triangulation are optimal



Input



Output

- LOP will generate uniquely determined triangulation of a set of points (using the PDDT criterion proposed in preferred-directions method)

Proposed Mesh-Coding Framework

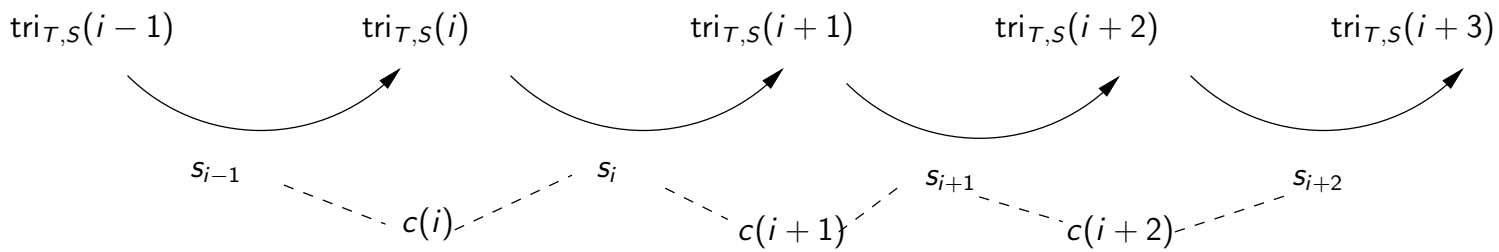
- A novel triangulation connectivity-coding mechanism developed and augmented to IT scheme for compressing mesh models of images
- Connectivity of triangulation expressed relative to connectivity of the uniquely-determined reference triangulation via sequence of edge flips
- Given a mesh model, encoding process performs following steps:
  1. mesh geometry (i.e., P and Z) coding using IT scheme (as described here)
  2. sequence generation
  3. sequence optimization (optional for encoding)
  4. sequence encoding
- Decoding process mirrors steps 1 and 4 above
- Connectivity can be progressively decoded (as well as images)
- Connectivity-coding process (i.e., steps 2-4) explained in more details

Step 2: Sequence Generation

- T: mesh's underlying triangulation
- T': uniquely-determined (optimal) triangulation
- Goal: generate sequence S of edge flips that transforms T' back to T as follows:
  1. Assign a unique integer label to each edge in T
  2. $T \rightarrow$ LOP optimization $\rightarrow$ sequence S' of edge flips $\rightarrow T'$
  3. Obtain S by reversing $S' \rightarrow S = \{s_i\}_{i=0}^{|S|-1}$
- Suspect edges for flipping are stored in a priority queue used in LOP
- Following priority schemes are considered:
  - first-in first-out (FIFO)
  - last-in first-out (LIFO)
  - lexicographic (i.e., edges are removed from queue in lexicographic order) (example given here)

Step 3: Sequence Optimization (Optional)

- Aims to improve locality by swapping elements of $S = \{s_i\}_{i=0}^{|S|-1}$
- Repeats passes over S until no swap is needed



- $c(i)$ estimates the cost between two edge flips s_{i-1} and s_i
- Given elements s_i and s_{i+1} in S in $\text{tri}_{T,S}(i)$, three decision functions proposed:

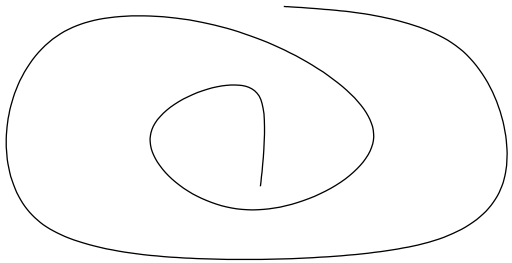
Rule 1: elements are swapped if this would reduce overall sum of relative indexes affected by the swap

Rule 2: elements are swapped if this would not increase cost of any of relative indexes affected by the swap AND at least one cost will be strictly reduced

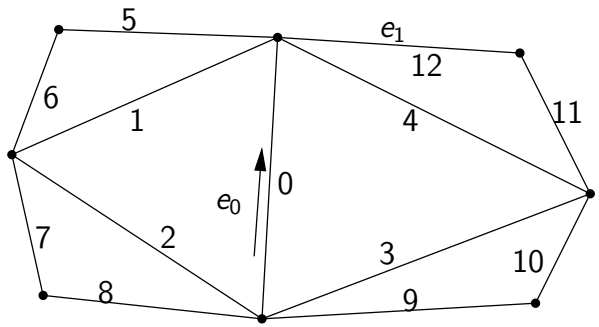
Rule 3: elements are swapped if this would strictly reduce ith relative index to be coded

Step 4: Sequence Encoding

- A relative indexing scheme (as described here) is employed
- Labels a subset of edges in triangulation relative to a particular edge



outward spiraling process



- Each element in edge-flip sequence S is coded relative to preceding one
- Arithmetic coding is used to code integer into bit-stream
- Effective approach as S tends to exhibit locality
- Provide progressive-coding functionality by coding S

Test Data

- 50 mesh models of images studied
- Mesh models grouped into two categories A and B based on their relative sequence lengths (i.e., $|S|$/number of edges in triangulation)
  - A: meshes have relative sequence length less than 15%
  - B: meshes have relative sequence length greater than 15%

Mesh Name	Image ϕ	Category	Vertex Count	Edge Count	Sequence $\|S\|$ Length†	Relative Sequence Length (%)†
A1	glasses2 (1024×768, 8 bps)	A	15728	47080	1070	2.3
A2	us (512×448, 8 bps)	A	6881	20632	1421	6.9
A3	wheel (512×512, 8 bps)	A	5242	15374	2298	14.9
A4	bull (1024×768, 8 bps)	A	7864	23536	410	1.7
B1	bull (1024×768, 8 bps)	B	7864	23536	7326	31.1
B2	glasses2 (1024×768, 8 bps)	B	7864	23514	6720	28.6
B3	lena (512×512, 8 bps)	B	2621	7797	3235	41.5
B4	lena (512×512, 8 bps)	B	2621	7797	2517	32.3

†Sequence generated with lexicographic priority scheme

Choice of Priority Scheme

Individual results

Dataset	Connectivity Coded Size (bits/vertex)		
	LIFO	FIFO	Lex.†
A1	1.02	1.02	**0.86**
A2	2.54	2.53	**1.98**
A3	4.01	3.95	**2.74**
A4	0.74	0.73	**0.61**
B1	8.36	8.42	**7.70**
B2	7.63	7.87	**7.24**
B3	9.81	9.93	**9.03**
B4	7.89	8.10	**7.64**

†lexicographic

- Lexicographic scheme beats other two schemes in all test cases
- Due to lexicographic scheme's ability to produce S with better locality

Choice of Decision Function

Individual results

Dataset	Connectivity Coded Size (bits/vertex)		
	Rule 1	Rule 2	Rule 3
A1	0.82	0.84	**0.76**
A2	1.92	1.93	**1.78**
A3	2.70	2.64	**2.54**
A4	0.59	0.59	**0.55**
B1	7.86	7.66	**7.44**
B2	7.43	7.20	**6.98**
B3	9.21	8.98	**8.67**
B4	7.71	7.58	**7.30**

- Rule 3 outperforms rules 1 and 2 in all test cases
- Due to rule 3's tendency to perform more swap operations, locality is improved more than other two rules

Proposed Method

- Proposed mesh-coding framework is most effective when employing
  - lexicographic priority scheme
  - decision function rule 3
- The proposed framework with preceding choices for free parameters is referred to as the **proposed method** for coding 2.5D mesh with arbitrary connectivity

Evaluation of Proposed Method

- Evaluated the proposed method by comparing it to a baseline scheme that is:
  - identical to our proposed method, except that
  - it encodes each edge label in edge-flip sequence as n-bit integer, where n is number of bits needed for integer representing edge labels

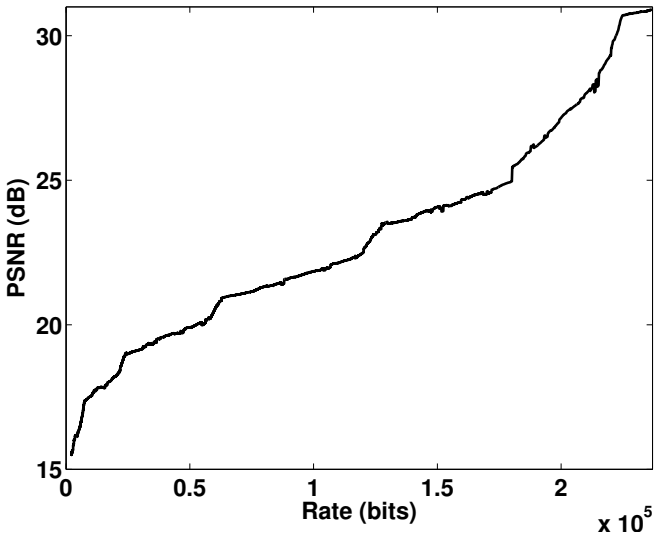
Individual results

Dataset	Connectivity Size (bits/vertex)		Total Size (bits/vertex)	
	Proposed	Baseline	Proposed	Baseline
A1	**0.76**	1.09	**15.07**	15.40
A2	**1.78**	3.11	**14.86**	16.19
A3	**2.54**	6.15	**11.38**	14.99
A4	**0.55**	0.79	**14.95**	15.19
B1	**7.44**	13.98	**21.84**	28.38
B2	**6.98**	12.83	**22.59**	28.44
B3	**8.67**	16.08	**24.28**	31.69
B4	**7.30**	12.51	**22.94**	28.15

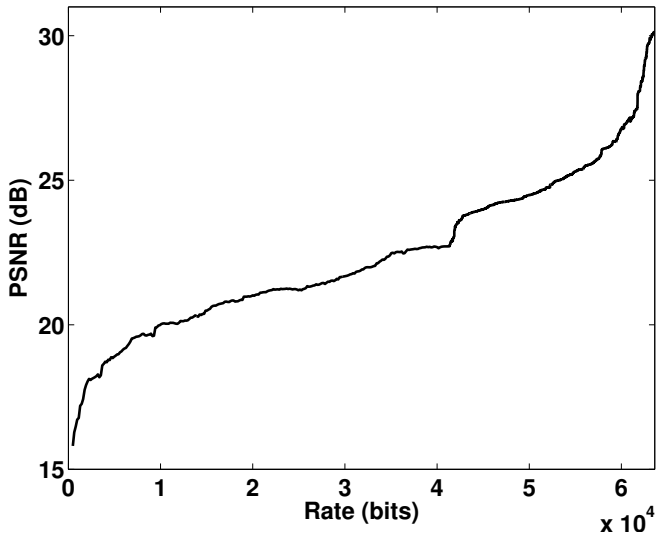
- In all 50 test cases, proposed method outperforms baseline scheme

Evaluation of proposed method (cont'd)

- For each mesh in A, proposed method is able to code mesh connectivity using less than 3.67 bits/vertex often cited for traditional approaches
- Proposed method has the progressive-coding functionality



mesh A1



mesh B3

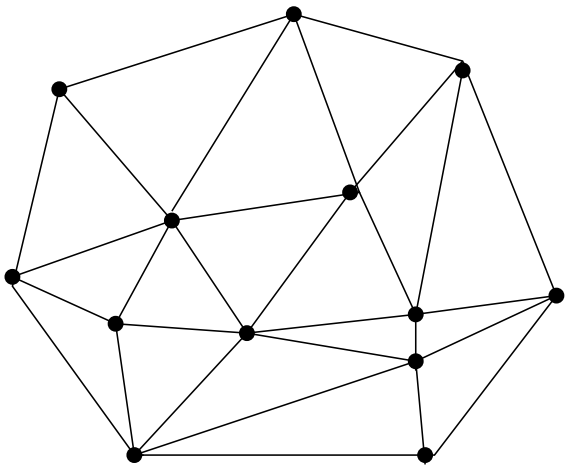
Conclusions

- Proposed novel coding mechanism for triangulation connectivity and its application to compressing mesh models of images
- Expresses connectivity of triangulation T a set P of points as edge-flip sequence that maps uniquely determined triangulation T' of P to T
- Through experiments, our proposed method consistently beats the baseline scheme
- Likely is able to outperform traditional connectivity coding methods for meshes that do not deviate too far from T'
- Can be beneficial to applications with functionality of progressive coding

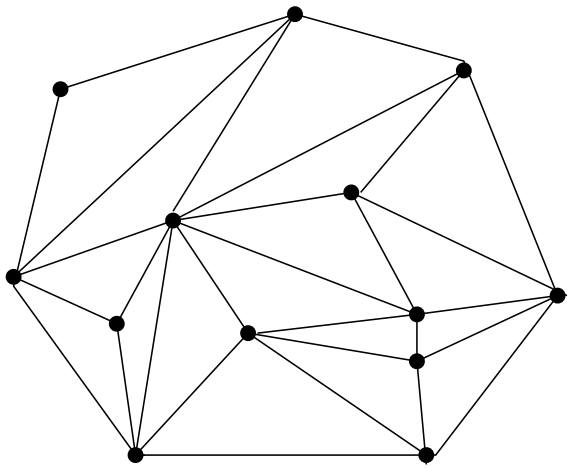
THANK YOU

Triangulation

- denote set of real numbers as $\mathbb{R}$
- **triangulation** T of set P of points in $\mathbb{R}^2$ is set of nondegenerate triangles such that:
  1. union of triangles in T is convex hull of P
  2. union of vertices of all triangles in T is P
  3. interiors of any two triangles in T are disjoint



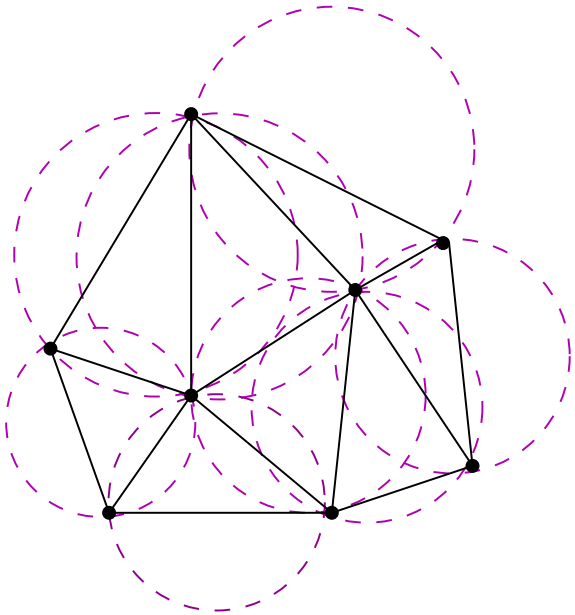
A triangulation of a point set P

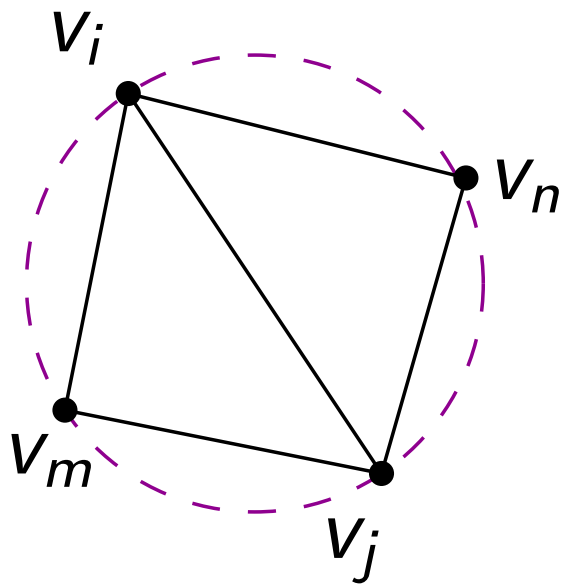


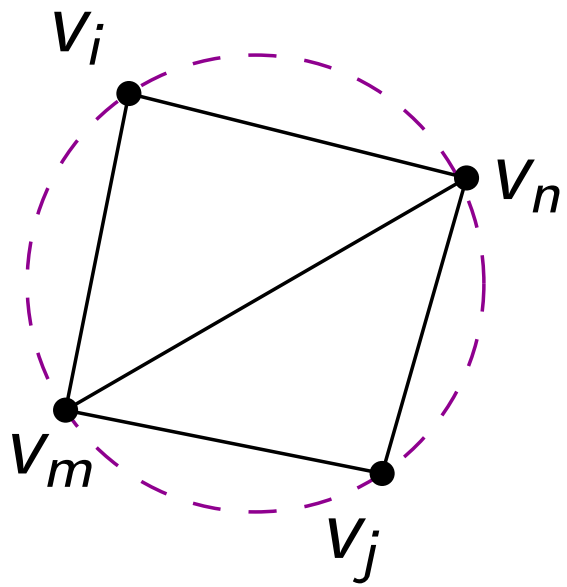
Another triangulation of P

Delaunay triangulation

- triangulation T of set P of points is **Delaunay** if no point in P is strictly inside the circumcircle of any triangle face of T
- Delaunay triangulation of set of points is not necessarily unique
- use preferred-directions method to yield unique (Delaunay) T of P, known as **preferred-directions Delaunay triangulation (PDDT)**





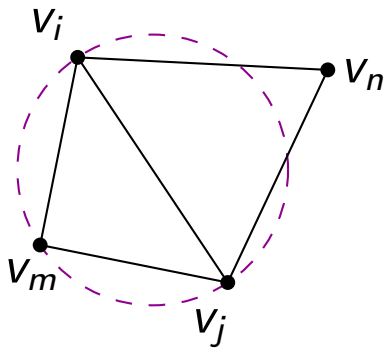


LOP (cont'd)

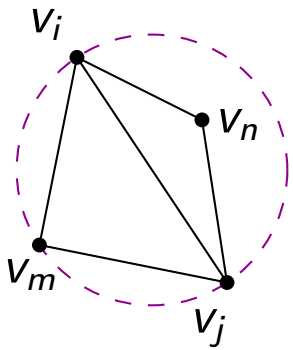
- algorithmically, LOP works as below:
  1. place all flippable edges in triangulation to suspect-edge queue (priority queue to store edges whose optimality is uncertain)
  2. perform following steps until suspect-edge queue is empty:
     1. remove e from front of suspect-edge queue;
     2. test e for optimality;
     3. if e not optimal, flip e and place any newly suspect edges (resulting from edge flip) on suspect-edge queue

The PDDT edge optimality criterion

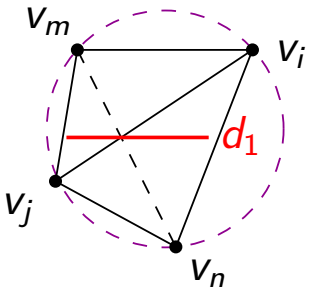
- optimality of edge $\overline{v_i v_j}$ is defined as follows



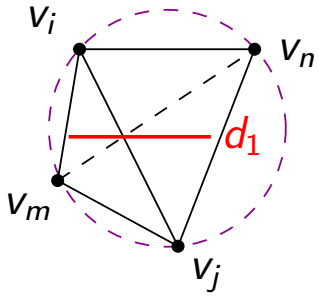
optimal



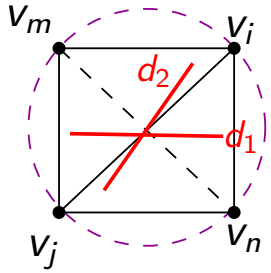
nonoptimal



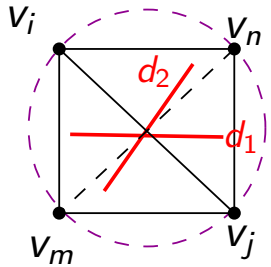
optimal



nonoptimal



optimal



nonoptimal

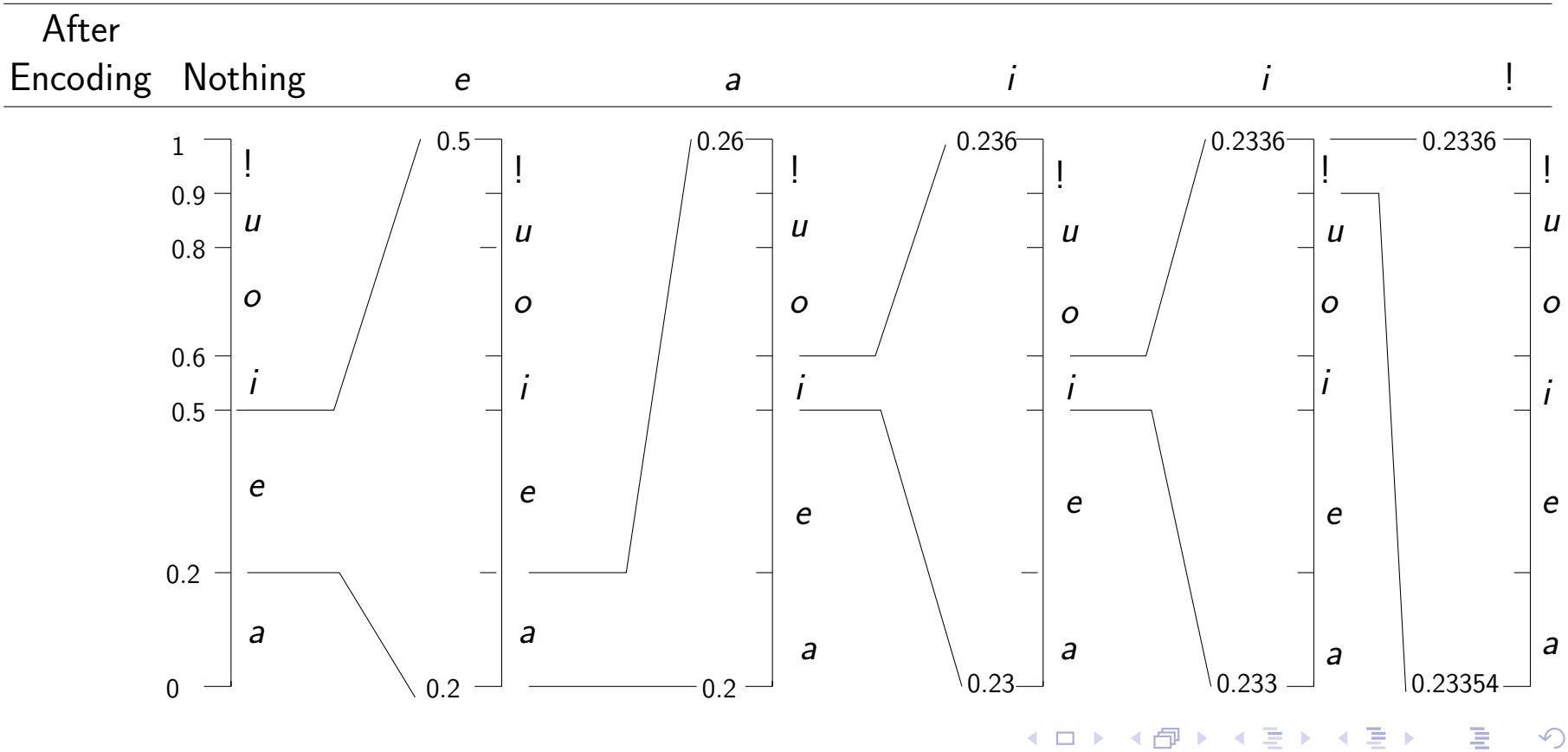
- in our work, $d_1 = (1, 0)$ and $d_2 = (1, 1)$ are utilized

Comments on progressive coding

- general term for compressing data, such as images or meshes
- best quality obtained if entire code stream decoded
- lower quality obtained if truncated version of code stream decoded
- information in code stream ordered by importance
- information about general trends comes early
- information about fine details comes later
- decoder constructs better data approximation as more of code stream received until entire code stream decoded

Arithmetic coding

- represent source message (i.e., a sequence of alphabets) as an interval between zero and one
- each successive symbol coded reduces size of interval in accordance with symbol's probability



Arithmetic coding (cont'd)

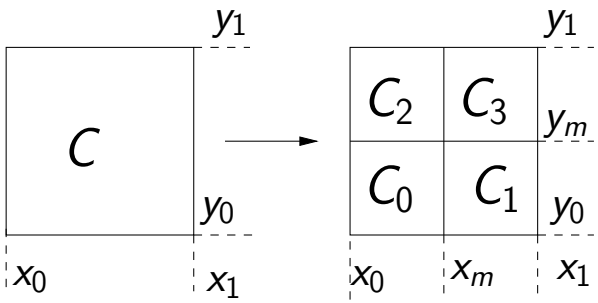
- binary arithmetic coder:
  - it only codes alphabets comprised of two symbols
  - if it need to code nonbinary symbol, binarization scheme (converting non-binary symbol to sequence of binary symbols) must be employed
- context-based arithmetic coder:
  - set of probabilities selected for coding symbol is based on contextual information (available to encoder and decoder), rather than always using same set of probabilities
- adaptive arithmetic coder
  - it updates probabilities of symbols to be coded during coding process

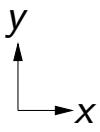
IT scheme

- assumes mesh connectivity uniquely determined by set of points
- based on a recursive quadtree partitioning of image domain along with iterative averaging process for sample data
- represents dataset using data structure called image tree (IT)
- employs arithmetic coder to efficiently code image tree
- provides progressive coding functionality by coding information in image tree nodes using top-down traversal of tree

IT scheme (cont'd): quadtree partitioning

- image domain $D = [0, W) \times [0, H)$ (i.e., width W and height H)
- $L-$level quadtree partitioning of D into rectangular regions called cells, where $L = \lceil \log_2 \max\{W, H\} \rceil + 1$
- root cell chosen as D and remaining cells formed by recursive splitting root cell to maximum of L levels
- cell C split in (approximately) half in each of horizontal and vertical directions to yield four child cells $\{C_i\}_{i=0}^{3}$ as shown below





$$c_0 = [x_0, x_m) \times [y_0, y_m) \qquad x_m = \left\lfloor \tfrac{1}{2}(x_0 + x_1 + 1) \right\rfloor$$
$$c_1 = [x_m, x_1) \times [y_0, y_m)$$
$$c_2 = [x_0, x_m) \times [y_m, y_1) \qquad y_m = \left\lfloor \tfrac{1}{2}(y_0 + y_1 + 1) \right\rfloor$$
$$c_3 = [x_m, x_1) \times [y_m, y_1)$$

- area of cell is number of lattice points it contains
- cell is empty if it contains no sample points
- cell is degenerate if it has zero area

IT scheme (cont'd): image tree

- tree based structure
- each node associated with nonempty cell from quadtree partitioning image domain and representative sample value z
- for leaf node, representative value chosen as sample value for corresponding point
- for nonleaf node, z is computed as approximate average of representative sample values $\{z_i\}_{i=0}^{N-1}$ of N child nodes as given by $z = \left\lfloor \frac{1}{N}\left(\sum_{i=0}^{N-1} z_i + \beta(N)\right)\right\rfloor$, where $\beta(n) = \lfloor n/2 \rfloor \, u(n-3)$

IT scheme (cont'd): image tree example

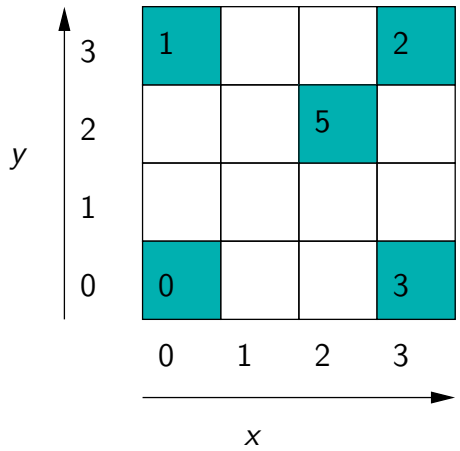
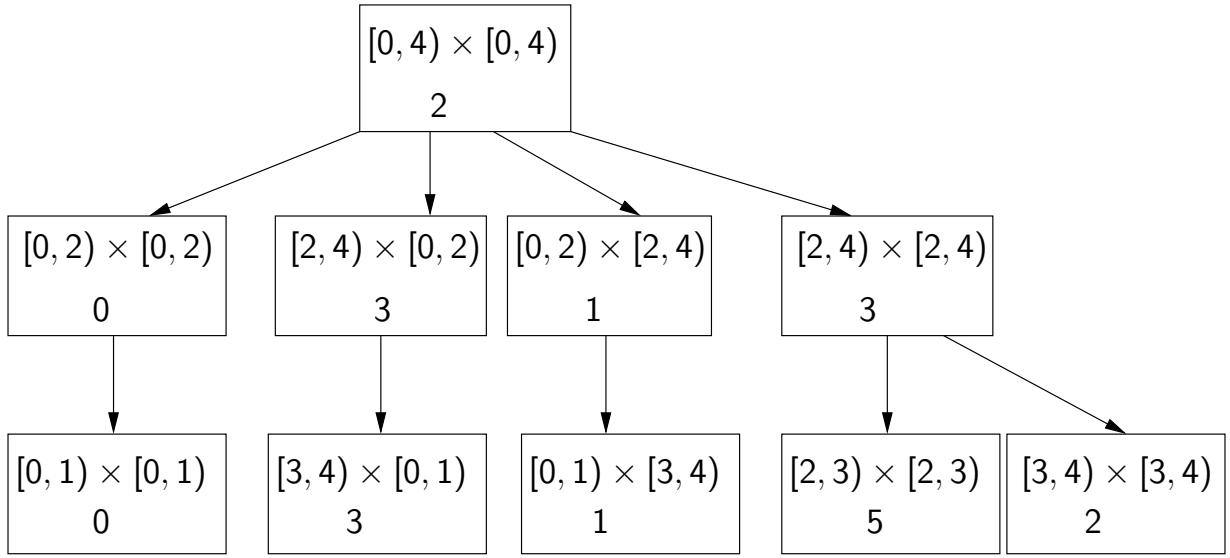


Image Dataset



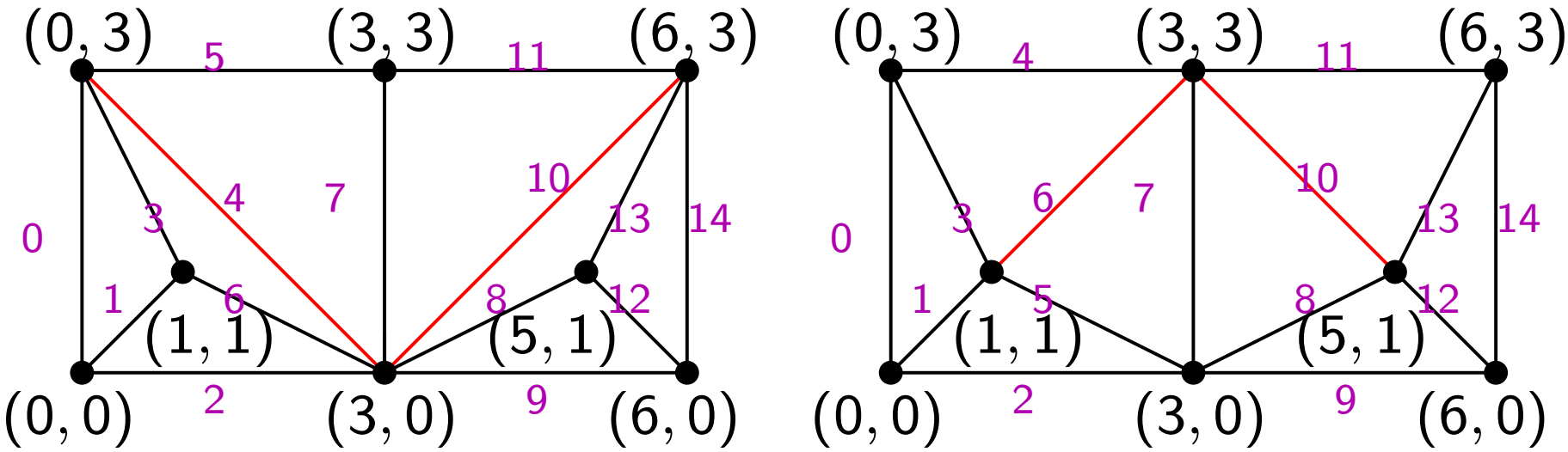
Corresponding Image Tree

- lossless reconstruct image dataset from information in leaf node
- can generate approximation dataset from pruned versions of tree
- one sample point and its corresponding sample value for each leaf node in pruned tree
- if leaf node has cell with area greater than one, corresponding sample point taken to be (approximate) centroid of cell

IT scheme (cont'd): coding of image tree

- image width W, image height H, P bits/sample
- algorithmically, coding process is as follows:
  1. Output W, H and P. Then, initialize arithmetic coding engine
  2. Clear the work queue (i.e., a priority queue for storing tree nodes)
  3. Encode representative sample value for root node
  4. Insert root node to the work queue
  5. If work queue is empty, stop. Otherwise, remove the next (i.e., highest priority) node n from the work queue
  6. If node n is nonleaf, encode node's child configuration (i.e., number of child nodes and cells with which child nodes are associated)
  7. For each of child nodes (if any), encode its representative sample value
  8. For each of child nodes (if any), compute its priority (for queue insertion) and insert it on the work queue
  9. Go to step 5

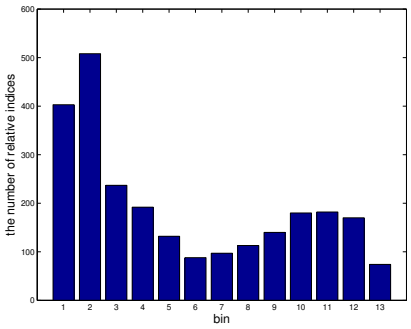
Step 2: sequence generation (cont'd)



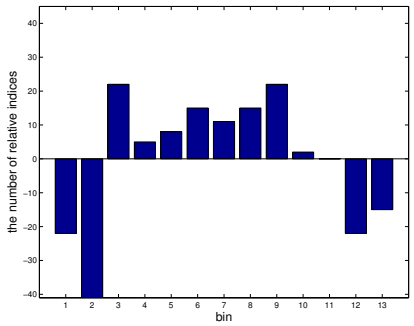
- apply LOP with PDDT edge-optimality criterion specified and keep record of edge flips in $S' = \{10, 4\}$
- obtain edge-flip sequence $S = \{6, 10\}$ by reversing S' and update each element label with respect to optimized triangulation

Choice of decision function (cont'd)

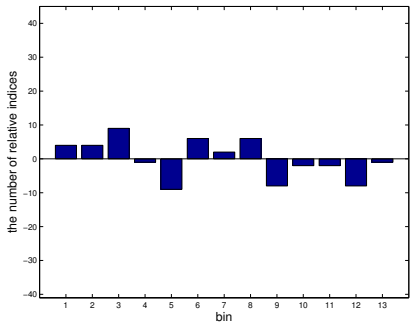
- provide insights into coding performance of each decision rule
- additional results for the test case of coding mesh B4 from earlier
- rule 1 reduced the number of very large and very small relative indexes
- rules 2 and 3 produced less very large and more very small relative indexes (compared to the approach without optimization)
- typical behaviors for other datasets as well



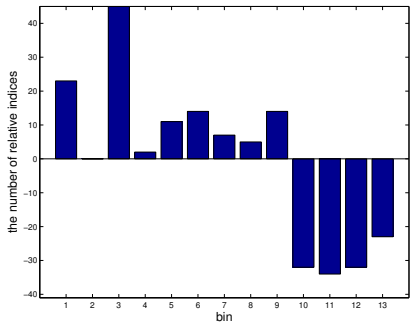
Without optimization



Decision function rule 1



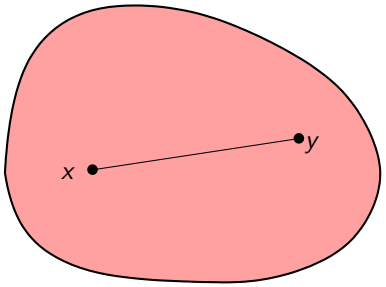
Decision function rule 2



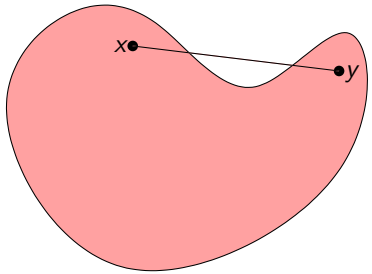
Decision function rule 3

Extra slides

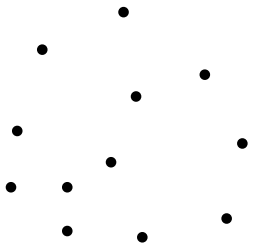
- convex set: in the point set X, for any pair of points x, y in X, every point on the line segment $\overline{xy}$ is in X;
- convex hull: the smallest convex set that contains point set X.
- degenerate triangle: collinear vertices, and zero area



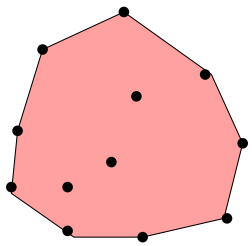
Convex set example



Non-convex set



A set X of points



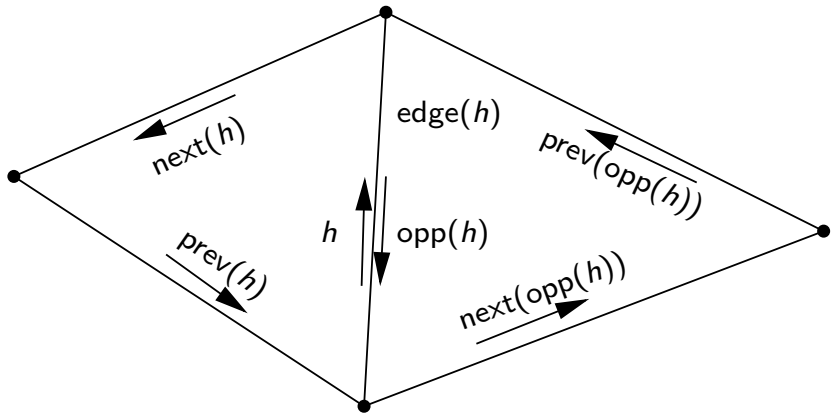
Convex hull of X

Extra slides (cont'd)

- convex polygon: all interior angles are less-than-or-equal-to 180°; strictly convex polygon: all interior angles are strictly less-than 180°
- for two line segments $\overline{a_0 a_1}$ and $\overline{b_0 b_1}$, we say that $\overline{a_0 a_1} < \overline{b_0 b_1}$ in lexicographic order if and only if either 1) $a'_0 < b'_0$, or 2) $a'_0 = b'_0$ and $a'_1 < b'_1$, where $a'_0 = \min(a_0, a_1)$, $a'_1 = \max(a_0, a_1)$, $b'_0 = \min(b_0, b_1)$, and $b'_1 = \max(b_0, b_1)$, and $\min(a, b)$ and $\max(a, b)$ denote the least and greatest of the points a and b in xy-lexicographic order, respectively
- denote $\mathrm{tri}_{T,S}$ as the triangulation obtained by applying all of the edge flips in S to T
- two edge-flip sequences S and S' are said to be **equivalent** if $\mathrm{tri}_{T,S} = \mathrm{tri}_{T,S'}$ (i.e., the application of each edge-flip sequence to T produces the same final triangulation)

Extra slides (cont'd)

- denote dirEdge(e) as the directed edge oriented from the smaller vertex to the larger vertex of edge e in xy-lexicographic order
- for a directed edge h: 1) opp(h) denotes the directed edge with the opposite direction as h; 2) next(h) and prev(h) denote the directed edges with the same left face as h that, respectively, follow and precede h in ccw order around their common left face; and 3) edge(h) denotes the (undirected) edge associated with h



Relative Indexing scheme: relIndex(e_1, e_0, τ, Θ) for sequence coding

1: $h :=$ dirEdge(e_0), mark all edges in τ as not visited, clear FIFO queue q, insert opp(h) and then h in q and $c := 0$
2: **while** q not empty **do**
3: remove element from front of q and set h to removed element
4: **if** edge(h) $\in \Theta$ and not visited **then**
5: **if** edge(h) $= e_1$ **then**
6: output c as index of edge e_1 relative to edge e_0 and stop
7: **endif**
8: mark edge(h) as visited and $c := c + 1$
9: **endif**
10: **if** opp(e) has left face and edge(next(opp(h))) not visited **then**
11: insert next(opp(h)) in q
12: **endif**
13: **if** opp(e) has left face and edge(prev(opp(h))) not visited **then**
14: insert prev(opp(h)) in q
15: **endif**
16: **endwhile**