Construction of 3D Orthogonal Cover of a Digital Object

N. Karmakar BESU Shibpur India

A. Biswas BESU Shibpur India

P. Bhowmick IIT Kharagpur India

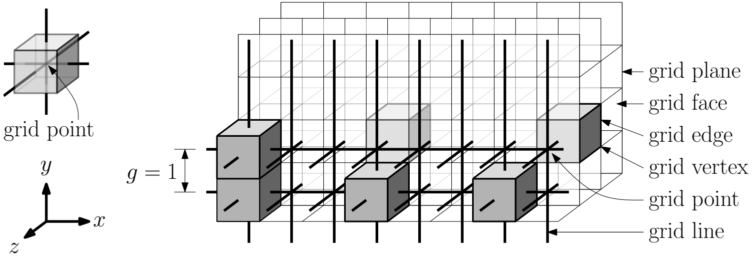
B. B. Bhattacharya ISI Kolkata India

IWCIA 2011

Madrid

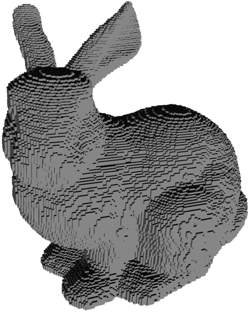
May 23–25, 2011

Theoretical Basis

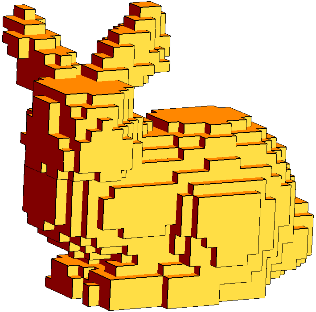


3D digital space and 26N [14].

Input & Output



$n = 58605$

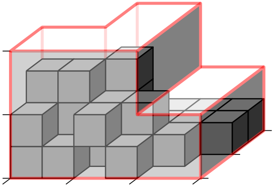


$g = 7 : n_v = 1031, n_f = 588$

Objective

Given the **3D object** A imposed on the **grid** $\mathbb{G}$, the problem is to construct its **orthogonal polytope** $P(A, \mathbb{G})$, such that:

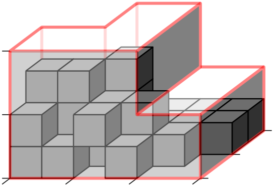
- each point $p \in A$ lies inside $P(A, \mathbb{G})$;
- each vertex of $P(A, \mathbb{G})$ is a grid vertex;
- each edge of $P(A, \mathbb{G})$ is parallel to one of the coordinate axes;
- each face of $P(A, \mathbb{G})$ lies on some grid plane;
- volume of $P(A, \mathbb{G})$ is minimized.



Objective

Given the **3D object** A imposed on the **grid** $\mathbb{G}$, the problem is to construct its **orthogonal polytope** $P(A, \mathbb{G})$, such that:

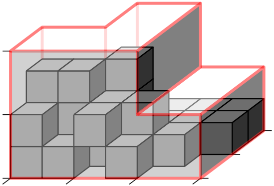
- each point $p \in A$ lies inside $P(A, \mathbb{G})$;
- each vertex of $P(A, \mathbb{G})$ is a grid vertex;
- each edge of $P(A, \mathbb{G})$ is parallel to one of the coordinate axes;
- each face of $P(A, \mathbb{G})$ lies on some grid plane;
- volume of $P(A, \mathbb{G})$ is minimized.



Objective

Given the **3D object** A imposed on the **grid** $\mathbb{G}$, the problem is to construct its **orthogonal polytope** $P(A, \mathbb{G})$, such that:

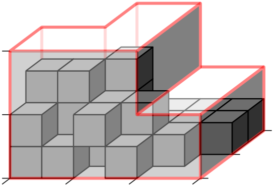
- each point $p \in A$ lies inside $P(A, \mathbb{G})$;
- each vertex of $P(A, \mathbb{G})$ is a grid vertex;
- each edge of $P(A, \mathbb{G})$ is parallel to one of the coordinate axes;
- each face of $P(A, \mathbb{G})$ lies on some grid plane;
- volume of $P(A, \mathbb{G})$ is minimized.



Objective

Given the **3D object** A imposed on the **grid** $\mathbb{G}$, the problem is to construct its **orthogonal polytope** $P(A, \mathbb{G})$, such that:

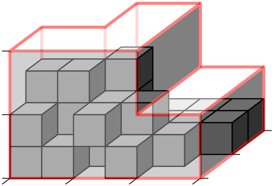
- each point $p \in A$ lies inside $P(A, \mathbb{G})$;
- each vertex of $P(A, \mathbb{G})$ is a grid vertex;
- each edge of $P(A, \mathbb{G})$ is parallel to one of the coordinate axes;
- each face of $P(A, \mathbb{G})$ lies on some grid plane;
- volume of $P(A, \mathbb{G})$ is minimized.



Objective

Given the **3D object** A imposed on the **grid** $\mathbb{G}$, the problem is to construct its **orthogonal polytope** $P(A, \mathbb{G})$, such that:

- each point $p \in A$ lies inside $P(A, \mathbb{G})$;
- each vertex of $P(A, \mathbb{G})$ is a grid vertex;
- each edge of $P(A, \mathbb{G})$ is parallel to one of the coordinate axes;
- each face of $P(A, \mathbb{G})$ lies on some grid plane;
- volume of $P(A, \mathbb{G})$ is minimized.



Data structures

- **Grid** $\mathbb{G}$: 3D array of UGCs (8 vertices, 12 edges, 6 faces)
- **Doubly connected edge list (DCEL)** [1, 18]:
  - **vertex list** $V = |vid|(x, y, z)|$
  - **edge list** $E = |eid|fid|start\ vid|next|prev|pair|$
  - **face list** $F = |fid|eid|plane|color|$

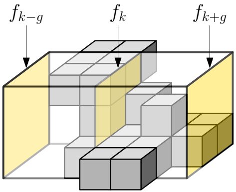
Data structures

- **Grid** $\mathbb{G}$: 3D array of UGCs (8 vertices, 12 edges, 6 faces)
- **Doubly connected edge list (DCEL)** [1, 18]:
  - **vertex list** $V = |vid|(x, y, z)|$
  - **edge list** $E = |eid|fid|start\ vid|next|prev|pair|$
  - **face list** $F = |fid|eid|plane|color|$

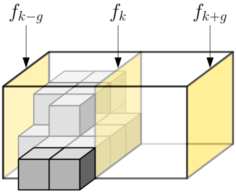
Data structures

- **Grid** $\mathbb{G}$: 3D array of UGCs (8 vertices, 12 edges, 6 faces)
- **Doubly connected edge list (DCEL)** [1, 18]:
  - **vertex list** $V = |vid|(x, y, z)|$
  - **edge list** $E = |eid|fid|start\ vid|next|prev|pair|$
  - **face list** $F = |fid|eid|plane|color|$

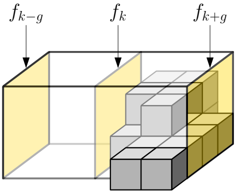
Eligibility of a face ($g = 3$)



$\text{ELIGIBLEFACE}(f_k) = \text{FALSE}$

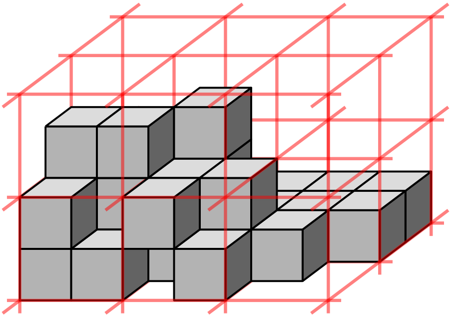


$\text{ELIGIBLEFACE}(f_k) = \text{TRUE}$

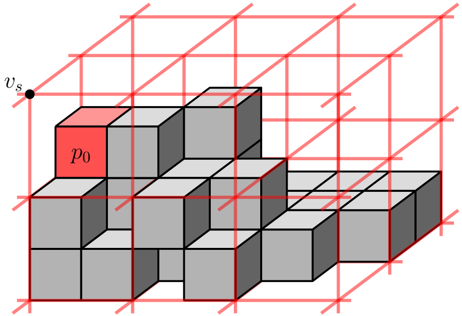


$\text{ELIGIBLEFACE}(f_k) = \text{TRUE}$

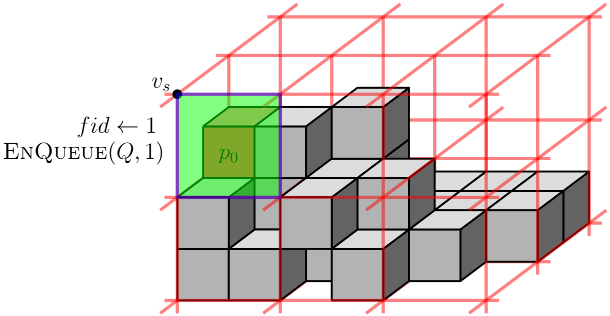
DCEL Construction



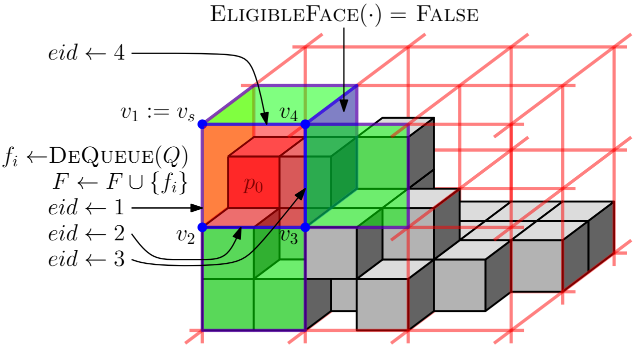
DCEL Construction



DCEL Construction



DCEL Construction



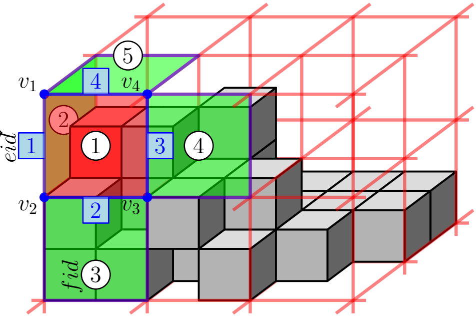
DCEL Construction

$$F = \{\ f_1 = \ |1|1|xy|\ \}$$

$$E = \left\{ \begin{array}{l} e_1 = \ |1|1|1|2|4|1|, \\ e_2 = \ |2|1|2|3|1|2|, \\ e_3 = \ |3|1|3|4|2|3|, \\ e_4 = \ |4|1|4|1|3|4| \end{array} \right\}$$

$$V = \{1, 2, 3, 4\}$$

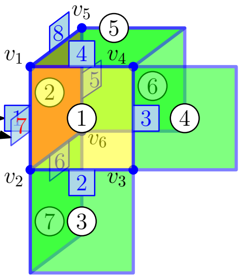
$$Q = \langle 2, 3, 4, 5 \rangle$$



f_1 dequeued

DCEL Construction

pairing of half-edges
$e_1(v_1, v_2) \in f_1$
and
$e_7(v_2, v_1) \in f_2$
not done for
non-coplanarity



f_2 dequeued

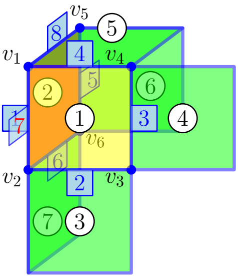
DCEL Construction

$$F = \left\{ \begin{array}{ll} f_1 = & |1|1|xy|, \\ \color{red}{f_2 =} & |2|5|yz| \end{array} \right\}$$

$$E = \left\{ \begin{array}{ll} e_1 = & |1|1|1|2|4|\color{red}{1}|, \\ e_2 = & |2|1|2|3|1|2|, \\ e_3 = & |3|1|3|4|2|3|, \\ e_4 = & |4|1|4|1|3|4|, \\ \color{red}{e_5 =} & |5|2|5|6|8|5|, \\ \color{red}{e_6 =} & |6|2|6|7|5|6|, \\ \color{red}{e_7 =} & |7|2|2|8|6|\color{red}{7}|, \\ \color{red}{e_8 =} & |8|2|1|5|7|8| \end{array} \right\}$$

$$V = \{1, 2, 3, 4, \color{red}{5, 6}\}$$

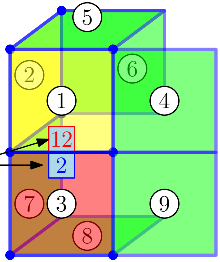
$$Q = \langle 3, 4, 5, \color{red}{6, 7} \rangle$$



f_2 dequeued

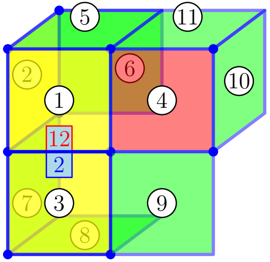
DCEL Construction

pairing of half-edges $e_2 \in f_1$ and $e_{12} \in f_3$, since f_1 and f_3 are coplanar



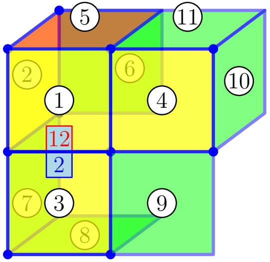
$Q = \langle 4, 5, 6, 7, 8, 9 \rangle$

DCEL Construction



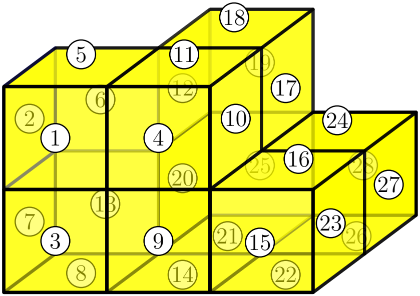
$Q = \langle 5, 6, 7, 8, 9, 10, 11 \rangle$

DCEL Construction



$$Q = \langle 6, 7, 8, 9, 10, 11 \rangle$$

DCEL Construction

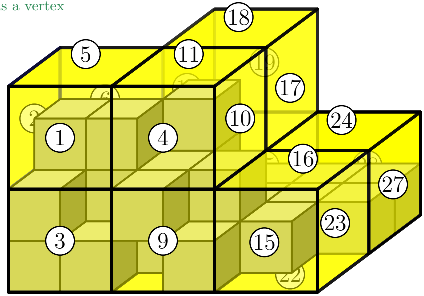


$Q = \langle \ \rangle$

DCEL Construction

Algorithm ORTHOCOVER3D$(A, \mathbb{G})$

$v_s \leftarrow (i_s = \lfloor i_0/g \rfloor \times g,\ j_s = \lceil j_0/g \rceil \times g,\ k_s = \lceil k_0/g \rceil \times g)$
pick the front face f_s of the UGC having v_s as a vertex
$n_f \leftarrow\ n_e \leftarrow n_v \leftarrow 0$
$fid[f_s] \leftarrow n_f \leftarrow n_f + 1$
$color[f_s] \leftarrow$ GRAY
ENQUEUE(Q, f_s)
while Q is not empty
  $f_i \leftarrow$ DEQUEUE(Q)
  for each edge $e_{ij} \in f_i$
    $eid[e_{ij}] \leftarrow n_e \leftarrow n_e + 1$
    $face[e_{ij}] \leftarrow fid[f_i]$
    if $start[e_{ij}] \notin V$
      $vid[start[e_{ij}]] \leftarrow n_v \leftarrow n_v + 1$
      $V \leftarrow V \cup \{start[e_{ij}]\}$
    assign $start[e_{ij}],\ next[e_{ij}],\ prev[e_{ij}]$
    ▷ from array $\mathbb{G}$
    $pair[e_{ij}] \leftarrow eid[e_{ij}]$
    $E \leftarrow E \cup \{e_{ij}\}$
    for each face f_k incident on
    $(start[e_{ij}], start[next[e_{ij}]])$
      if ELIGIBLEFACE(f_k) = TRUE
        if $color[f_k]$ = WHITE
          $fid[f_k] \leftarrow n_f \leftarrow n_f + 1$
          $color[f_k] \leftarrow$ GRAY
          EnQueue(Q, f_k)
        else if $color[f_k]$ = BLACK **and** f_i and f_k are coplanar
          $pair[e_{ij}] \leftarrow eid[\bar{e}_{ij}], pair[\bar{e}_{ij}] \leftarrow eid[e_{ij}]$ ▷ $\bar{e}_{ij} \in f_k$
  $edge[f_i] \leftarrow eid[e_{i1}]$ ▷ e_{i1} is the 1st edge of f_i
  $color[f_i] \leftarrow$ BLACK
  $F \leftarrow F \cup \{f_i\}$
MERGEFACE(F, E)



Face Merging

starting with f_1:

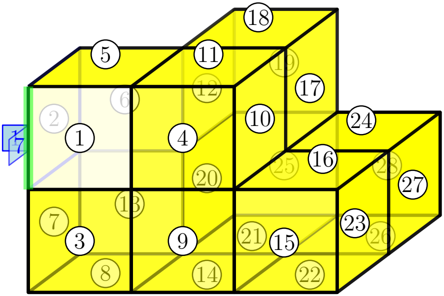
init: $count \leftarrow 4$;

f_1 not merged with f_2,

as $pair[e_1] = e_1$;

so $e_2 = next[e_1]$ and

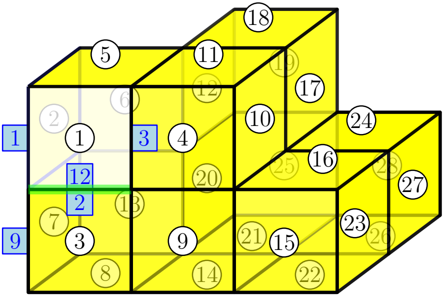
$count \leftarrow count - 1 = 3$.



Face Merging

f_3 merged with f_1,
as $pair[e_2] = e_{12} \neq e_2$;
$next[prev[e_2]] =\leftarrow next[e_{12}]$,
or, $next[e_1] \leftarrow e_9$;
$next[prev[e_{12}]] \leftarrow next[e_2]$,
or, $next[e_{11}] \leftarrow e_3$;
$face[e_{12}] \leftarrow f_1$;
f_3 deleted from F;
e_2, e_{12} deleted from E;
$count \leftarrow count + 2 = 5$;

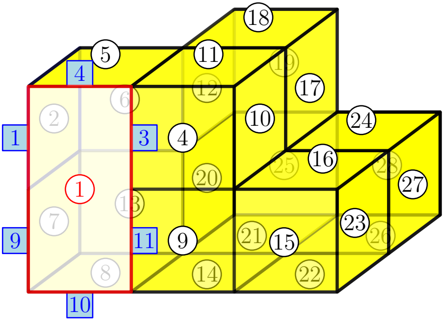
repeat with next edge e_9
until $count = 0$.



Face Merging

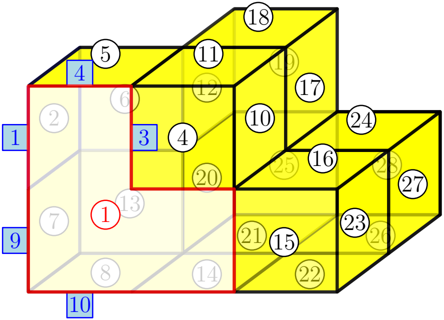
f_3 merged with f_1,
as $pair[e_2] = e_{12} \neq e_2$;
$next[prev[e_2]] = \leftarrow next[e_{12}]$,
or, $next[e_1] \leftarrow e_9$;
$next[prev[e_{12}]] \leftarrow next[e_2]$,
or, $next[e_{11}] \leftarrow e_3$;
$face[e_{12}] \leftarrow f_1$;
f_3 deleted from F;
e_2, e_{12} deleted from E;
$count \leftarrow count + 2 = 5$;

repeat with next edge e_9
  until $count = 0$.



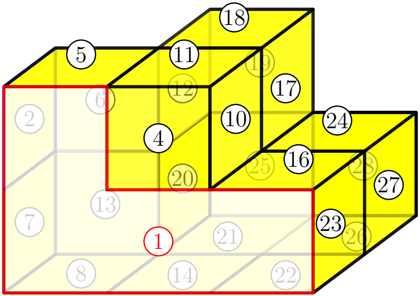
Face Merging

f_7 not merged with f_1,
$count \leftarrow count - 1 = 4$;
f_8 not merged with f_1,
$count \leftarrow count - 1 = 3$;
f_9 merged with f_1,
$count \leftarrow count + 2 = 5$;



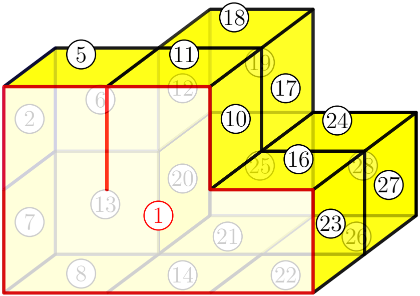
Face Merging

f_{14} not merged with f_1,
$count \leftarrow count - 1 = 4$;
f_{15} merged with f_1,
$count \leftarrow count + 2 = 6$;



Face Merging

f_{22} not merged with f_1,
$count \leftarrow count - 1 = 5$;
f_{23} not merged with f_1,
$count \leftarrow count - 1 = 4$;
f_{16} not merged with f_1,
$count \leftarrow count - 1 = 3$;
f_4 merged with f_1,
$count \leftarrow count + 2 = 5$;



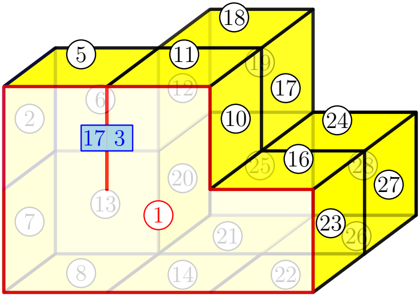
Face Merging

f_{10} not merged with f_1,
$count \leftarrow count - 1 = 4$;
f_{11} not merged with f_1,
$count \leftarrow count - 1 = 3$;

$face[e_{17}] = face[e_3]$,
$count \leftarrow count - 1 = 2$;

$face[e_3] = face[e_{17}]$,
$count \leftarrow count - 1 = 1$;

f_5 not merged with f_1,
$count \leftarrow count - 1 = 0$;



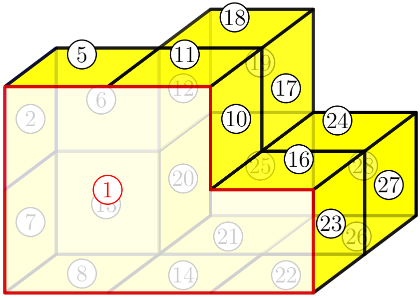
Face Merging

f_{10} not merged with f_1,
$count \leftarrow count - 1 = 4$;
f_{11} not merged with f_1,
$count \leftarrow count - 1 = 3$;

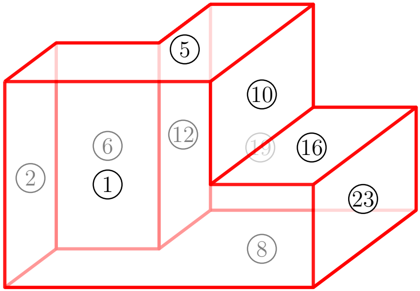
$face[e_{17}] = face[e_3]$,
$count \leftarrow count - 1 = 2$;

$face[e_3] = face[e_{17}]$,
$count \leftarrow count - 1 = 1$;

f_5 not merged with f_1,
$count \leftarrow count - 1 = 0$;



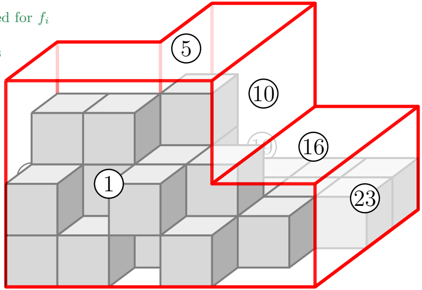
Face Merging



Face Merging

Procedure MergeFace(F, E)

01. **for** each face $f_i \in F$
02. $e_{ij} \leftarrow edge[f_i]$ ▷ 1st edge of $f_i \in F$
03. $count \leftarrow 4$ ▷ number of edges to be visited for f_i
04. **do**
05. **if** $eid[e_{ij}] \neq pair[e_{ij}]$ ▷ $pair[e_{ij}]$ exists
06. $\bar{e}_{ij} \leftarrow pair[e_{ij}]$
07. $f_k \leftarrow face[\bar{e}_{ij}]$
08. $next[prev[e_{ij}]] \leftarrow next[\bar{e}_{ij}]$
09. $next[prev[\bar{e}_{ij}]] \leftarrow next[e_{ij}]$
10. $e_t \leftarrow next[prev[e_{ij}]]$
11. $face[\bar{e}_{ij}] \leftarrow fid[f_i]$
12. $F \leftarrow F \setminus \{f_k\}$
13. $E \leftarrow E \setminus \{e_{ij}, \bar{e}_{ij}\}$
14. $e_{ij} \leftarrow e_t$
15. $edge[f_i] \leftarrow eid[e_{ij}]$
▷ 1st edge
of the modified face f_i
16. $count \leftarrow count + 2$
▷ 4 edges added to
& 2 edges removed from f_i
17. **else** ▷ $pair[e_{ij}]$ does not exist
18. $e_{ij} \leftarrow next[e_{ij}]$
19. $count \leftarrow count - 1$
20. **while** $count \neq 0$
21. **return** (F, E)



Time Complexity

For each UGC, at most five faces.
DCEL complexity (#faces, #edges, #vertices) $\leqslant O(n/g^3)$.

Best Case:
Eligibility of a face $= O(1)$ time.
Min #UGCs $= O(n/g^3)$.
$\Rightarrow$ DCEL time $= O(n/g^3) \times O(1) = \mathbf{O(n/g^3)}$.
Max #faces on the surface $O(n/g^3)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g^3)}$.

Worst Case:
Eligibility of a face $= O(g^3)$ time.
Max #UGCs $= O(n/g)$.
$\Rightarrow$ DCEL time $= O(n/g) \times O(g^3) = \mathbf{O(ng^2)}$.
Maximum number of faces $= O(n/g)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g)}$.

In practice: Tends towards the **best case**.

Time Complexity

For each UGC, at most five faces.
DCEL complexity (#faces, #edges, #vertices) $\leqslant O(n/g^3)$.

Best Case:
Eligibility of a face $= O(1)$ time.
Min #UGCs $= O(n/g^3)$.
$\Rightarrow$ DCEL time $= O(n/g^3) \times O(1) = \mathbf{O(n/g^3)}$.
Max #faces on the surface $O(n/g^3)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g^3)}$.

Worst Case:
Eligibility of a face $= O(g^3)$ time.
Max #UGCs $= O(n/g)$.
$\Rightarrow$ DCEL time $= O(n/g) \times O(g^3) = \mathbf{O(ng^2)}$.
Maximum number of faces $= O(n/g)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g)}$.

In practice: Tends towards the **best case**.

Time Complexity

For each UGC, at most five faces.
DCEL complexity (#faces, #edges, #vertices) $\leqslant O(n/g^3)$.

Best Case:
Eligibility of a face $= O(1)$ time.
Min #UGCs $= O(n/g^3)$.
$\Rightarrow$ DCEL time $= O(n/g^3) \times O(1) = \mathbf{O(n/g^3)}$.
Max #faces on the surface $O(n/g^3)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g^3)}$.

Worst Case:
Eligibility of a face $= O(g^3)$ time.
Max #UGCs $= O(n/g)$.
$\Rightarrow$ DCEL time $= O(n/g) \times O(g^3) = \mathbf{O(ng^2)}$.
Maximum number of faces $= O(n/g)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g)}$.

In practice: Tends towards the **best case**.

Time Complexity

For each UGC, at most five faces.
DCEL complexity (#faces, #edges, #vertices) $\leqslant O(n/g^3)$.

Best Case:
Eligibility of a face $= O(1)$ time.
Min #UGCs $= O(n/g^3)$.
$\Rightarrow$ DCEL time $= O(n/g^3) \times O(1) = \mathbf{O(n/g^3)}$.
Max #faces on the surface $O(n/g^3)$.
$\Rightarrow$ MERGEFACE time $= \mathbf{O(n/g^3)}$.

Worst Case:
Eligibility of a face $= O(g^3)$ time.
Max #UGCs $= O(n/g)$.
$\Rightarrow$ DCEL time $= O(n/g) \times O(g^3) = \mathbf{O(ng^2)}$.
Maximum number of faces $= O(n/g)$.
$\Rightarrow$ MERGEFACE time $= \mathbf{O(n/g)}$.

In practice: Tends towards the **best case**.

Time Complexity

For each UGC, at most five faces.
DCEL complexity (#faces, #edges, #vertices) $\leqslant O(n/g^3)$.

Best Case:
Eligibility of a face $= O(1)$ time.
Min #UGCs $= O(n/g^3)$.
$\Rightarrow$ DCEL time $= O(n/g^3) \times O(1) = \mathbf{O(n/g^3)}$.
Max #faces on the surface $O(n/g^3)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g^3)}$.

Worst Case:
Eligibility of a face $= O(g^3)$ time.
Max #UGCs $= O(n/g)$.
$\Rightarrow$ DCEL time $= O(n/g) \times O(g^3) = \mathbf{O(ng^2)}$.
Maximum number of faces $= O(n/g)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g)}$.

In practice: Tends towards the **best case**.

Time Complexity

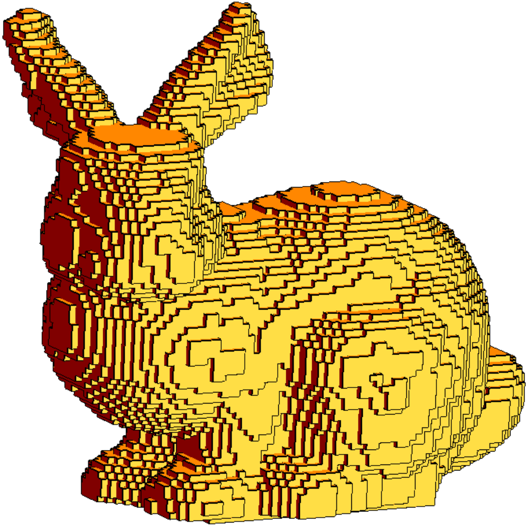
For each UGC, at most five faces.
DCEL complexity (#faces, #edges, #vertices) $\leqslant O(n/g^3)$.

Best Case:
Eligibility of a face $= O(1)$ time.
Min #UGCs $= O(n/g^3)$.
$\Rightarrow$ DCEL time $= O(n/g^3) \times O(1) = \mathbf{O(n/g^3)}$.
Max #faces on the surface $O(n/g^3)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g^3)}$.

Worst Case:
Eligibility of a face $= O(g^3)$ time.
Max #UGCs $= O(n/g)$.
$\Rightarrow$ DCEL time $= O(n/g) \times O(g^3) = \mathbf{O(ng^2)}$.
Maximum number of faces $= O(n/g)$.
$\Rightarrow$ MergeFace time $= \mathbf{O(n/g)}$.

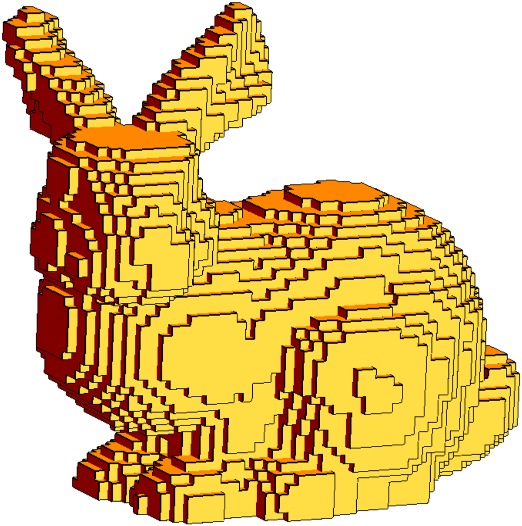
In practice: Tends towards the **best case**.

Results: Stanford bunny



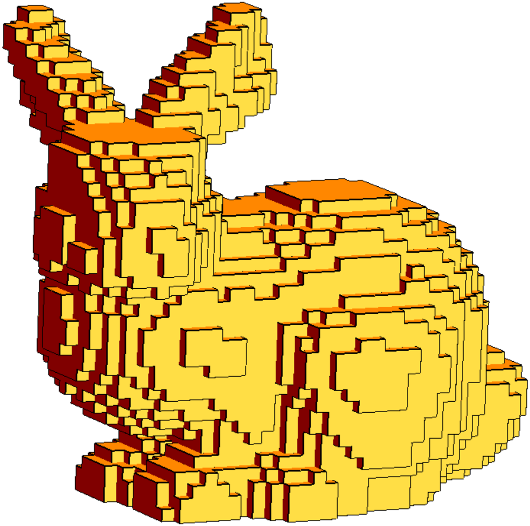
$g = 2$
$n_v = 9090$
$n_e = 43458$
$n_f = 5529$

Results: Stanford bunny



$g = 3$
$n_v = 3987$
$n_e = 19134$
$n_f = 2367$

Results: Stanford bunny



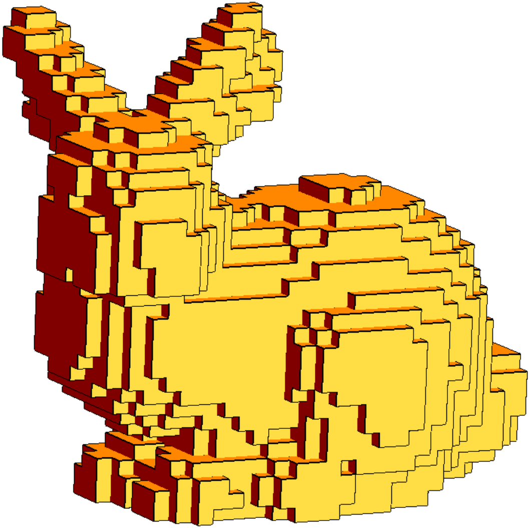
$g = 4$

$n_v = 2335$

$n_e = 11164$

$n_f = 1383$

Results: Stanford bunny



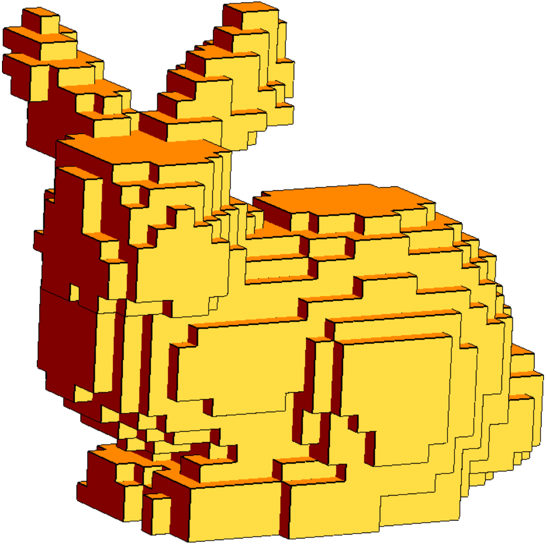
$g = 5$

$n_v = 1533$

$n_e = 7358$

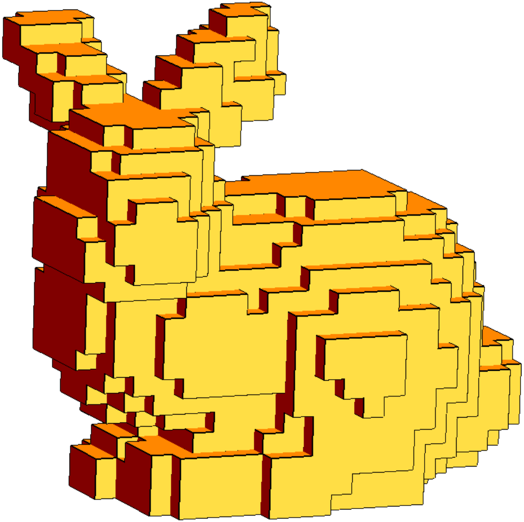
$n_f = 895$

Results: Stanford bunny



$g = 6$
$n_v = 1031$
$n_e = 5042$
$n_f = 588$

Results: Stanford bunny



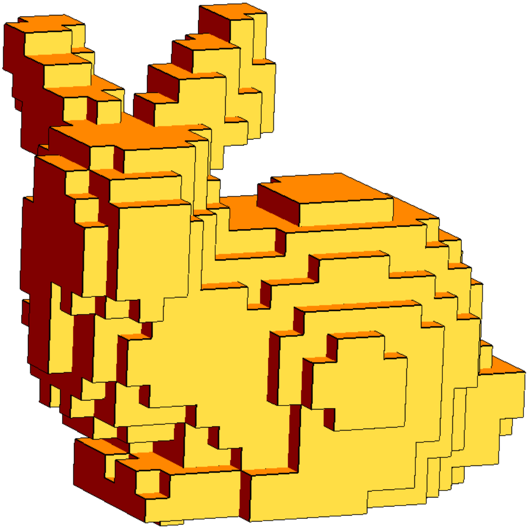
$g = 7$

$n_v = 775$

$n_e = 3894$

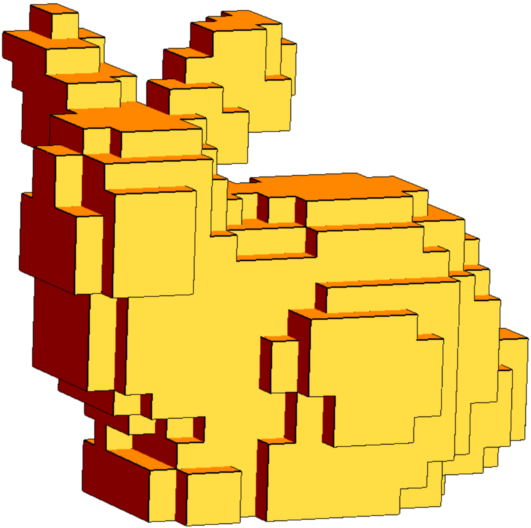
$n_f = 455$

Results: Stanford bunny



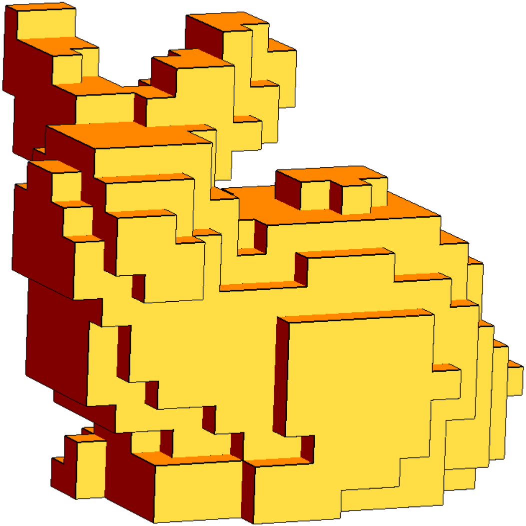
$g = 8$
$n_v = 636$
$n_e = 2972$
$n_f = 359$

Results: `Stanford bunny`



$g = 9$
$n_v = 502$
$n_e = 2440$
$n_f = 287$

Results: Stanford bunny



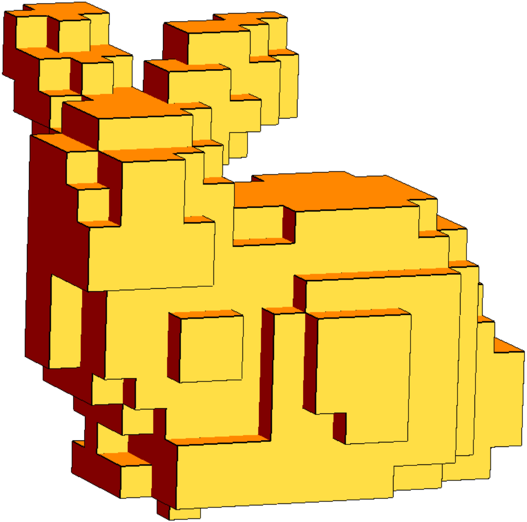
$g = 10$

$n_v = 426$

$n_e = 2016$

$n_f = 243$

Results: Stanford bunny



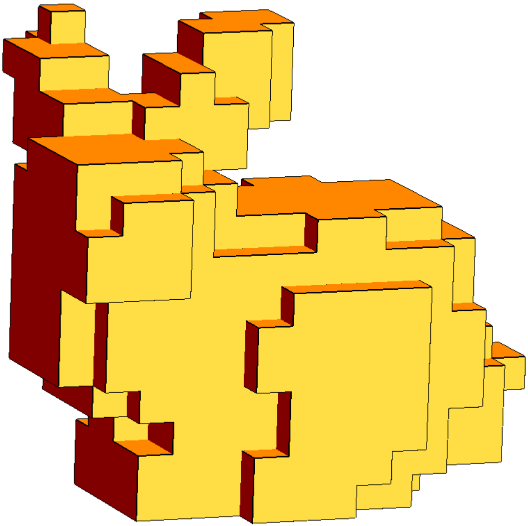
$g = 11$

$n_v = 340$

$n_e = 1696$

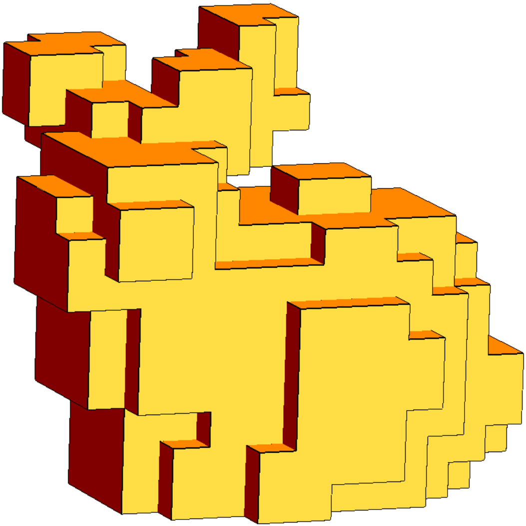
$n_f = 204$

Results: Stanford bunny



$g = 12$
$n_v = 296$
$n_e = 1398$
$n_f = 173$

Results: Stanford bunny



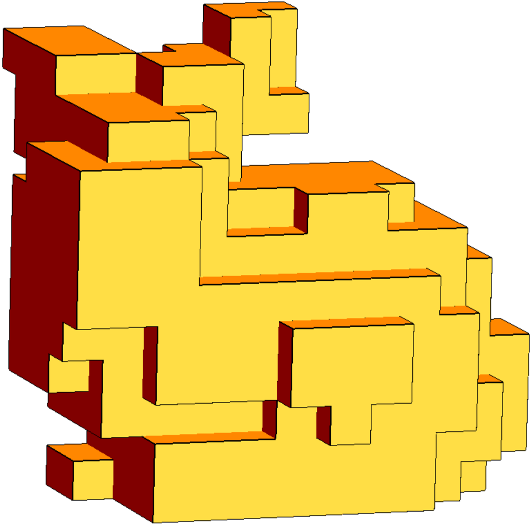
$g = 13$

$n_v = 245$

$n_e = 1246$

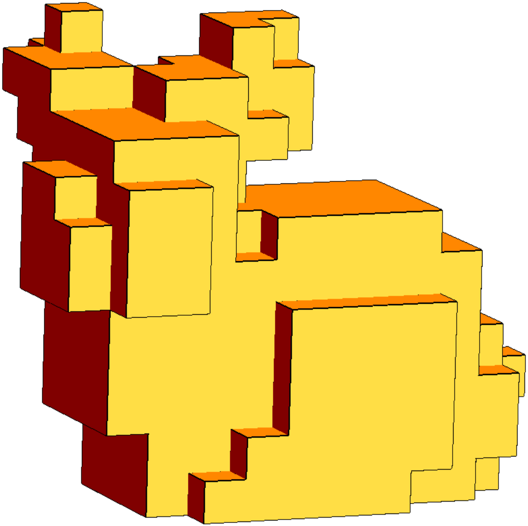
$n_f = 140$

Results: `Stanford bunny`



$g = 14$
$n_v = 210$
$n_e = 1094$
$n_f = 125$

Results: `Stanford bunny`



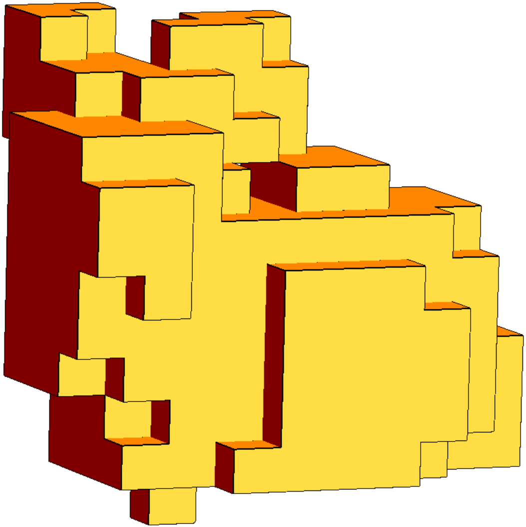
$g = 15$

$n_v = 192$

$n_e = 948$

$n_f = 118$

Results: Stanford bunny



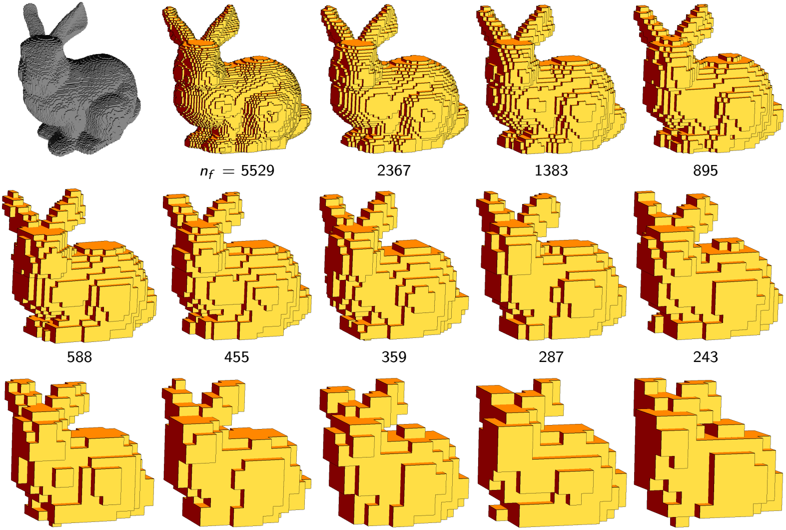
$g = 16$

$n_v = 179$

$n_e = 856$

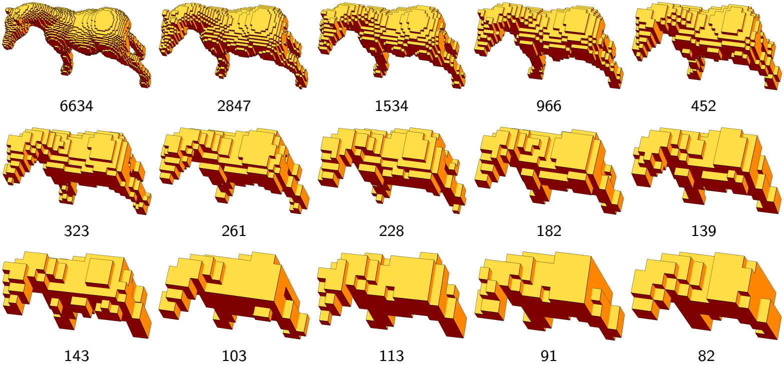
$n_f = 105$

Results: Stanford bunny

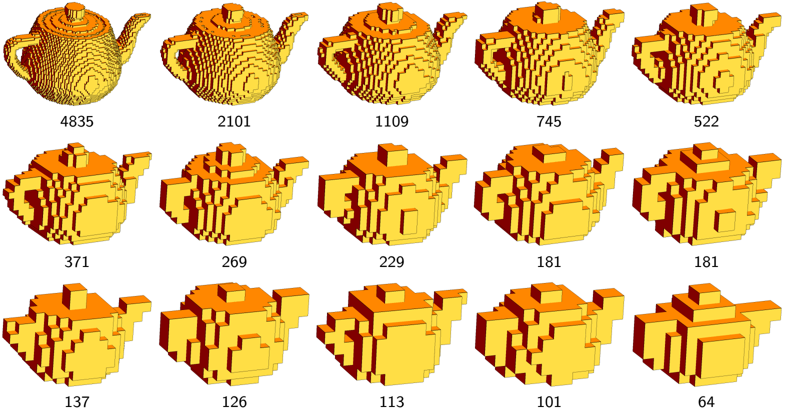


n_f = 5529, 2367, 1383, 895, 588, 455, 359, 287, 243, 204, 173, 140, 125, 105

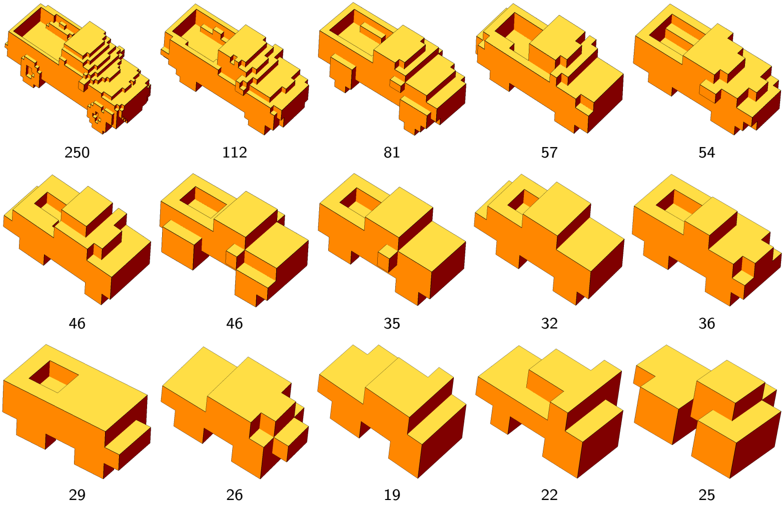
Results: horse ($g = 2, 3, \ldots, 16$)



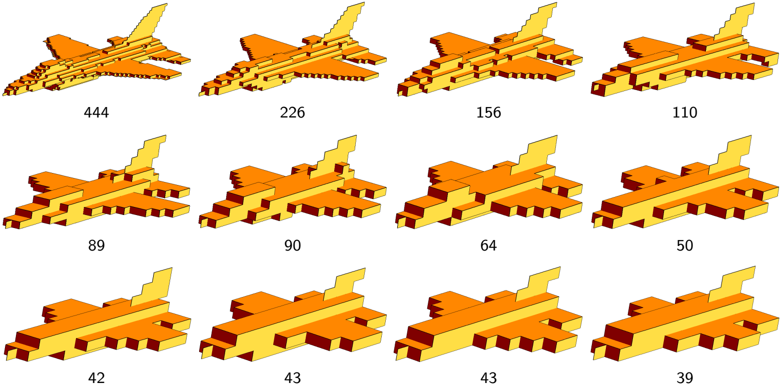
Results: teapot ($g = 2, 3, \ldots, 16$)



Results: `pickup van` ($g = 2, 3, \ldots, 16$)



Results: `airplane` $(g = 2, 3, \ldots, 13)$



References I

Berg, M.D., Cheong, O., Kreveld, M.V., Overmars, M.: Computational Geometry—Algorithms and Applications. Springer-Verlag, Berlin Heidelberg (1997)

Bhaniramka, P., Wenger, R., Crawfis, R.: Isosurface construction in any dimension using convex hulls. IEEE Trans. on Visualization and Computer Graphics, Vol. 10, pp. 130–41 (2004)

Bhaniramka, P., Wenger, R., Crawfis, R.: Isosurfacing in higher dimensions. In Proceedings of Visualization, Salt Lake City, pp. 267–73 (2000)

Biswas, A., Bhowmick, P., Bhattacharya, B.B.: Construction of Isothetic Covers of a Digital Object: A Combinatorial Approach. Journal of Visual Communication and Image Representation, Vol. 21, pp. 295–310 (2010)

References II

Biswas, A., Bhowmick, P., Bhattacharya, B.B.: TIPS: On Finding a Tight Isothetic Polygonal Shape Covering a 2D Object. 14th Scandinavian Conference on Image Analysis, Joensuu, Finland, LNCS, Vol. 3540, pp. 930–939 (2005)

Brimkov, V.: Discrete volume polyhedrization: Complexity and bounds on performance. in: Computational Methodology of Objects Represented in Images: Fundamentals, Methods and Applications, Proc. of the International Symposium CompIMAGE '06, pp. 117–122., Taylor and Francis, Coimbra, Portugal (2006)

Coeurjolly, D., Sivignon, I.: Reversible discrete volume polyhedrization using Marching Cubes simplification. SPIE Vision Geometry XII, 5300 (1–11) (2004)

Cohen-Or, D., Shamir, A., Shapira, L.: Consistent Mesh Partitioning and Skeletonization using the Shape Diameter Function. The Visual Computer, Vol. 24, pp. 249–259 (2008)

References III

Giles, M., Haimes, R.: Advanced interactive visualization for CFD. Computing Systems in Engineering, Vol. 1, pp. 51–62 (1990)

Golovinskiy, A., Funkhouser, T.: Randomized cuts for 3D mesh analysis. ACM Transactions on Graphics (Proc. SIGGRAPH ASIA), Vol. 27, Article 145 (2008)

Hearn, D., Baker, M.P.: Computer Graphics with OpenGL, Pearson Education Inc. (2004)

Hill, F.S., Kelley, S.M.: Computer Graphics Using OpenGL, Pearson Edcation Inc. (2007)

Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. The Visual Computer, Vol. 21, pp.649–658 (2005)

Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Picture Analysis. Morgan Kaufmann, San Francisco (2004)

References IV

Livnat, Y., Shen, H-W, Johnson, C.: A near optimal isosurface extraction algorithm using span space. IEEE Trans. Visualization and Computer Graphics, Vol. 2, pp. 73–84 (1996)

Lorensen, W.E., Cline, H.E.: Marching Cubes: A high resolution 3D surface construction algorithm. Computer Graphics, Vol. 21, pp. 163–169 (1987)

Newman, T.S., Yi, H.: A Survey of the Marching Cubes Algorithm. Computers & Graphics, Vol. 30, pp. 854–879 (2006)

Preparata, F.P., Shamos, M.I.: Computational Geometry—An Introduction. Spinger-Verlag, New York (1985)

Shapira, L., Shalom, S., Shamir, A., Cohen-Or, D., and Zhang H.: Contextual Part Analogies in 3D Objects. International Journal of Computer Vision, Vol. 89, pp. 309–326 (2010)

Shlafman, S., Tal, A., Katz, S.: Metamorphosis of Polyhedral Surfaces using Decomposition, Eurographics 2002, pp. 219–228

References V

- Turk, G., Levoy, M.: Zippered polygon meshes from range images. SIGGRAPH 1994, pp. 311-318.
- Weber, G., Kreylos, O., Ligocki, T., Shalf, J., Hagen, H., Hamann, B.: Extraction of crack-free isosurfaces from adaptive mesh refinement data. In: Proceedings of VisSym '01, Ascona, Switzerland, pp. 25–34 (2001)
- Wilhelms, J., van Gelder, A.: Topological considerations in isosurface generationâextended abstract. Computers Graphics, Vol. 24, pp. 79–86 (1990)

Thank You