Partitioning 3D Phantoms Into Homogeneous Cuboids *

Anuj Jain & Sartaj Sahni
Department of Computer and Information Science and Engineering
University of Florida, Gainesville, FL 32611
{ajain, sahni}@cise.ufl.edu
Jatinder Palta & James Dempsey
Department of Radiation Oncology
University of Florida, Gainesville, FL 32611
{paltajr,dempsey}@ufl.edu

Abstract

We analyze the heuristic proposed by Jung [3] to partition a 3D phantom into homogeneous cuboids. We show that the 2D version of this heuristic generates the minimum number of rectangles when partitioning 2D non-degenerate phantoms. However, this 2D version doesn't generate optimal partitions of degenerate 2D phantoms. In fact, the heuristic may generate more than 3 times as many rectangles as is necessary. The 3D heuristic of [3] doesn't generate optimal partitionings of 3D simple phantoms either. We show that slicing partitions of 3D simple phantoms are optimal. Our analysis suggests a new heuristic for 3D phantoms. This heuristic generated one-third as many cuboids when used to partition a 3D CT-scan phantom and about one-fourth as many cuboids on randomly generated 3D phantoms as produced by the heuristic of [3].

Keywords: 3D phantom partitioning, cuboids, heuristic.

1 Introduction

Two- and three-dimensional phantoms[1] are often partitioned into homogeneous rectangles and cuboids, respectively, so as to simplify algorithms that operate on the phantom [4]–[7]. For example, in radiation treatment planning, the patient is described by a 3D phantom [4, 3]. This 3D phantom is composed of unit cubes/voxels, where different voxels may have different density (so, the phantom is a collection of heterogeneous voxels). Dose computation is done via ray tracing and the dose computation algorithms require the computation of the radiological distance between the end points of the ray. The radiological distance, which is the Euclidean distance weighted by the density, may be obtained by tracing the path of the ray from one voxel to the next and aggregating the density-weighted Euclidean distance. A faster way to compute this distance is to trace the path of the ray examining only points at which the ray moves

*This work was supported, in part, by the National Library of Medicine under grant LM06659-03.
[1]A phantom is a heterogeneous rectangle or cuboid.

from a voxel of one density to a voxel of a different density. To facilitate this latter faster ray trace, Li and Williamson propose a collection of 3D convex shapes that may be used to partition a 3D phantom into a collection of homogeneous shapes. However, the decomposition is to be done manually. Jung [3] proposes an automatic decomposition into cuboids that have the same density (i.e., into homogeneous cuboids). We use 2DPPP (3DPPP) as an abbreviation for the 2D (3D) phantom partitioning problem. In 2DPPP (3DPPP) we wish to partition a non-homogeneous 2D (3D) phantom into a minimum number of homogeneous rectangles (cuboids).

The region partitioning problem is closely related to the phantom partitioning problem. In the 2D (3D) region partitioning problem, we are given a connected collection of homogeneous pixels (voxels) and are to partition these into a minimum number of rectangles (cuboids).

The 2D region partitioning problem (2DRPP), has been quite well studied [2, 6, 5]. For example, Ohtsuki [6] has shown that the minimum number of non-overlapping rectangles in the solution to a 2DRPP is $n/2 + h - d - 1$, where n is the number of vertices in the region, h is the number of holes, and d is the maximum number of independent degenerate chords (these terms are defined in Section 3). Imai and Asano [2] develop an $O(n^{3/2} \log n)$ time algorithm for 2DRPP, and Nahar and Sahni [5] develop an $O(n + v \log v)$, where v is the number of vertical inversions, time algorithm for 2DRPP instances that have no holes. Wu and Sahni [7] develop algorithms for the case when we are to find a minimum number of (possibly overlapping) rectangles to cover the region. 3DRPP is known to be NP-hard [1].

Jung [3] has proposed a heuristic to partition a 3D phantom into a minimum number of homogeneous cuboids. In this paper, we analyze this heuristic. Specifically, we show that the 2D version of this heuristic is optimal for 2D non-degenerate phantoms. (Section 3), and the 3D version is non-optimal even for simple 3D phantoms (Section 4). When the heuristic is used on degenerate 2D phantoms, it may produce partitions that have more than three times as many rectangles as is necessary. When used on general 3D phantoms, the number of cuboids produced by the heuristic of [3] may be the square (or more) of the minimum number of cuboids needed to partition the phantom.

Our analysis of Jung's [3] heuristic leads to a new heuristic (Section 6), which on all test phantoms resulted in far fewer cuboids than did Jung's [3] heuristic (Section 7).

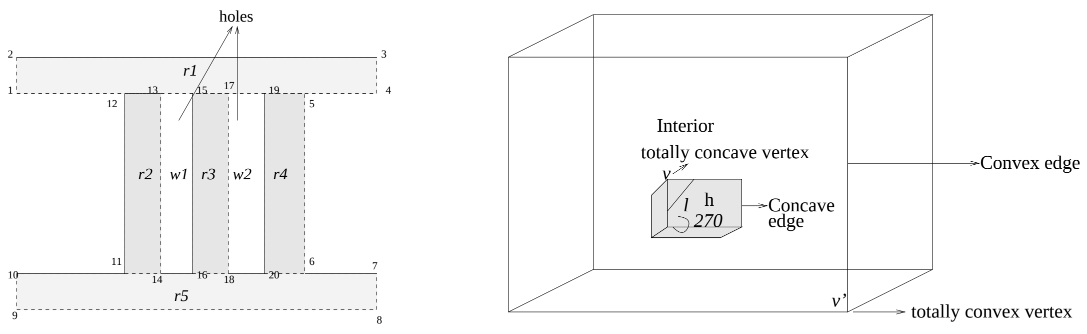
2 Preliminaries

Definition 1 *A* pixel *is a unit square whose sides are parallel to the x- and y-axes. A* voxel *is a unit cube whose sides are parallel to the x-y, y-z and x-z planes. Each pixel/voxel is homogeneous and has* *a non-negative integer density associated with it. A 2D (3D)* phantom *is a collection of pixels (voxels) that defines a rectangle (cuboid). In 2DPPP (3DPPP), we are to partition the pixels (voxels) of a 2D (3D) phantom into a minimum number of homogeneous rectangles (cuboids), where the pixels (voxels) in each homogeneous rectangle (cuboid) have the same density. A* component *of a phantom is a maximal connected collection of pixels/voxels that have the same density.*

Definition 2 *A* region *is a connected collection of pixels/voxels that have the same density. The pixels/voxels that comprise a region define the* interior *of the region. The remaining pixels/voxels define the* exterior *of the region. In* 2*DRPP (*3*DRPP), we are to partition the interior pixels (voxels) of the 2D (3D) region into a minimum number of rectangles (cuboids).*

Instances of 2DRPP and 3DRPP may be drawn by either shading the interior pixels/voxels as in Figure 1(a) (in this case, exterior pixels are unshaded) or by shading the exterior pixels as in Figure 1(b). In these drawings, individual pixels (voxels) are not shown; rather the boundaries of interior/exterior regions are drawn as lines or edges (faces) that are parallel to the x- and y-axes (xy-, yz-, xz-planes). These regions are *rectilinear regions.*

Definition 3 *A* hole *is a connected set of exterior pixels/voxels that is bounded by interior pixels/voxels on all sides. Figure 1(a) shows 2D holes (w1 and w2 are holes), and Figure 1(b) shows a 3D hole (the shaded cuboid is a hole).*



(a) 2D region (b) 3D region

Figure 1: Example 2D and 3D holes

An instance of 2DPPP (3DPPP) may be solved by decomposing a phantom into its unique components (note that each component defines a region) and solving the 2DRPP (3DRPP) defined by each of these components. Hence, 2DPPP may be solved optimally using the 2DRPP algorithms of [2, 6, 5]. Further, from the proof of [1], it follows that 3DPPP is NP-hard.

Jung's [3] heuristic (1) for 3DPPP partitions the phantom into cuboids in two steps–initial cuboid construction and cuboid growing.

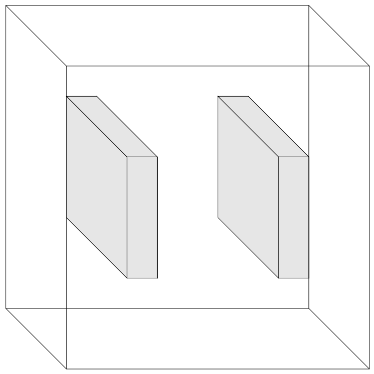
Heuristic 1 *Jung's [3] heuristic.*

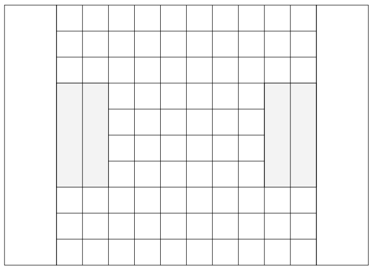
1. **Initial Cuboid Construction** *In this step the given heterogeneous phantom is partitioned into a set of (homogeneous) cuboids. We start with each voxel defining a cuboid. Then, pairs of adjacent cuboids that have the same size and density are combined. This pairwise combining of adjacent cuboids continues until no more combining is possible. The search for combinable pairs is done in the following way. First examine adjacent pairs of cuboids along the x-axis, then along the y-axis, then along the z-axis. This three-stage examination is repeated until no adjacent cuboids that have the same size and density remain. The cuboids resulting from this step are called* unexamined cuboids.

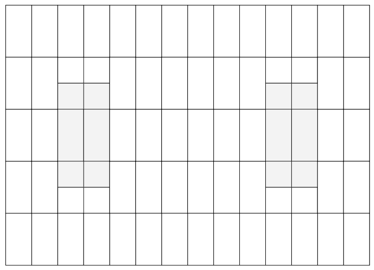
2. **Cuboid Growing** *In this step the number of cuboids is decreased by extending each of the unexamined cuboids as much as possible. The cuboids from Step 1 are examined in decreasing order of volume. The faces of the cuboid being examined are moved outward, in decreasing order of face area, till a face with different density or a face of an already examined cuboid or the boundary of the phantom is reached. When extending a face, previously unexamined, cuboids with the same density are either fully or partially absorbed into the growing cuboid. When an unexamined cuboid is partly absorbed into the growing cuboid, the rest of the unexamined cuboid is split into new cuboids that are labeled as unexamined cuboids. A cuboid once examined and grown is* finalized *and is not subject to further change. Cuboid growing continues until no unexamined cuboids remain.*

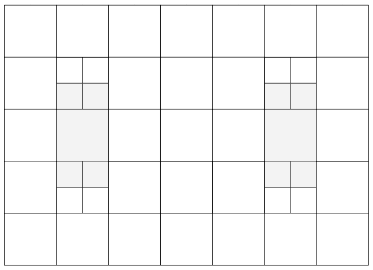
A 2D version of Heuristic 1 may be obtained by eliminating the work done in one of the three dimensions, say dimension z.

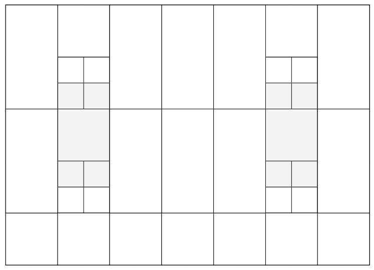
We use the 3D phantom of Figure 2(a) to illustrate the working of Jung's heuristic. Figure 2(b) gives a 2D view of the phantom. The 2D view is obtained by projecting the 3D view on to the front face of the phantom. The 2D view is used for ease of drawing. Figures 2(c)–(g) show the initial cuboid construction step of Jung's algorithm. Figures 3(a) and (b) show the 2 iterations of step 2 of Jung's heuristic that take place on our example. Figure 3(c) shows the end results and Figure 3(d) shows the 3D view of the parititioning.

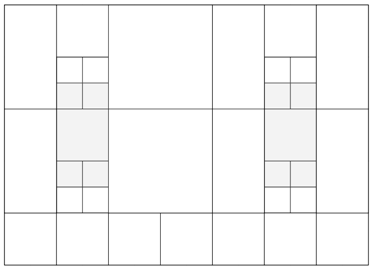

(a) 3D Phantom region


(b) 2D View


(c) Step 1


(d) Step 1


(e) Step 1


(f) Step 1

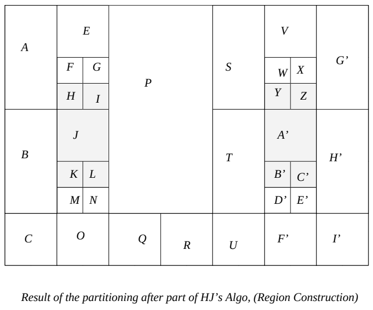
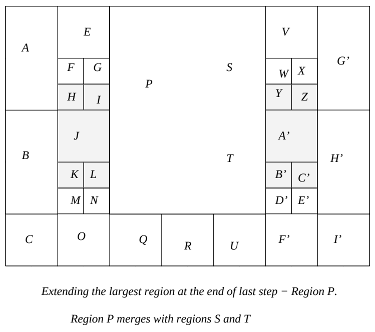
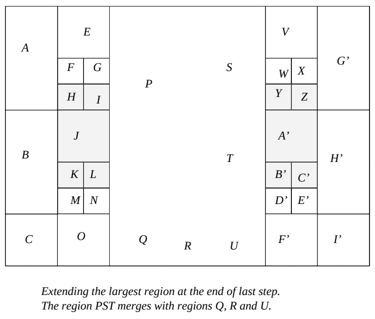

A
E
F
G
H
I
J
K
L
M
N
B
C
O
P
Q
R
S
T
U
V
W
X
Y
Z
A'
B'
C'
D'
E'
F'
G'
H'
I'
Result of the partitioning after part of HJ's Algo, (Region Construction)
(g) Step 1

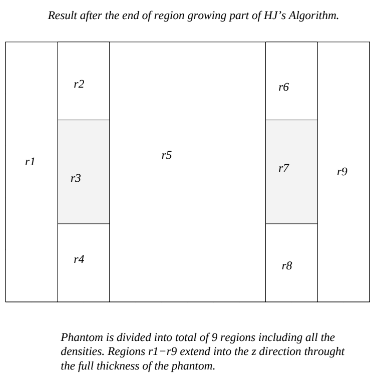
Figure 2: Step 1 of Jung's heuristic


A
E
F
G
H
I
P
S
V
W
X
Y
Z
G'
B
J
K
L
M
N
T
A'
B'
C'
D'
E'
H'
C
O
Q
R
U
F'
I'
Extending the largest region at the end of last step − Region P.
Region P merges with regions S and T

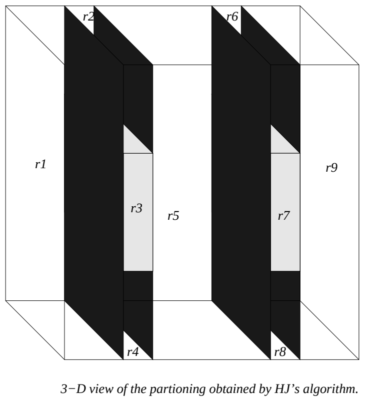
(a) Step 2


A
E
F
G
H
I
P
S
V
W
X
Y
Z
G'
B
J
K
L
M
N
T
A'
B'
C'
D'
E'
H'
C
O
Q
R
U
F'
I'
Extending the largest region at the end of last step.
The region PST merges with regions Q, R and U.

(b) Step 2


Result after the end of region growing part of HJ's Algorithm.
r1
r2
r3
r4
r5
r6
r7
r8
r9
Phantom is divided into total of 9 regions including all the densities. Regions r1−r9 extend into the z direction throught the full thickness of the phantom.

(c) Final 2D



(d) Final 3D

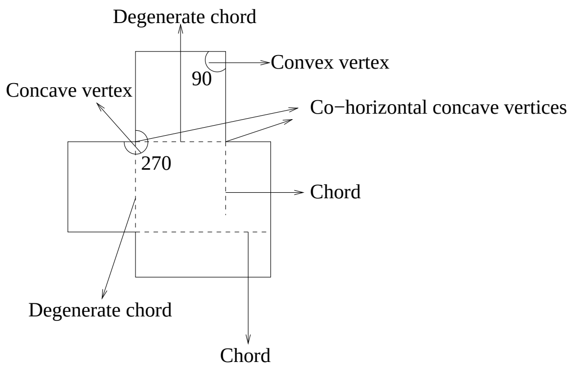
Figure 3: Step 2 of Jung's heuristic

3 Analysis for 2D Phantoms (2DPPP)

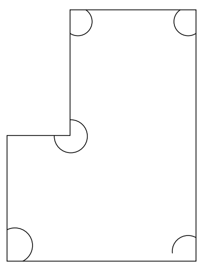
In this section, we determine the conditions under which the 2D version of Heuristic 1 obtains a partitioning into a minimum number of rectangles. Since, pixels of one component of the phantom cannot be in the same rectangle as pixels of another component, it is sufficient to analyze Heuristic 1 for an instance of 2DRPP in which the interior pixels are to be partitioned into a minimum number of rectangles. Before proceeding with the analysis, we define several terms.

Definition 4 Final result–*The rectangles into which the interior pixels of the 2DRPP instance are partitioned by Heuristic 1.*

Definition 5 Concave vertex–*A vertex in a rectilinear 2D region such that the two edges incident on the vertex make an interior angle of 270 degrees (Figure 4(a)).*
Convex vertex–*A vertex in a rectilinear 2D region such that the two edges incident on the vertex make an interior angle of 90 degree (Figure 4(a)).*



(a) Degenerate region



(b) Non-degenerate region

Figure 4: Degenerate and non-degenrate regions

Definition 6 Chord–*A horizontal or a vertical line segment that begins at a concave vertex, extends in the region interior, and ends in the region interior or at the first point at which it meets an edge of the region (Figure 4(a)).*
Degenerate chord–*A chord that joins two co-horizontal (having the same y-coordinate) or two co-*

vertical (having the same x-coordinate) concave vertices. (Figure 4(a)).
Two degenerate chords are **independent** *iff they do not have a common vertex.*

Definition 7 Degenerate rectilinear region–*A rectilinear region is degenerate iff it is possible to draw a degenerate chord in the region (Figure 4(a)).* **Non-degenerate rectilinear region**–*A rectilinear region in which it isn't possible to draw a degenerate chord is called non-degenerate (Figure 4(b)).*
A 2D phantom is **degenerate** *iff it has a degenerate component. A 2D phantom is* **non-degenerate** *iff it has no degenerate component.*

The following lemmas are used to prove that Heuristic 1 partitions a non-degenerate 2D region optimally.

Lemma 1 *The sides of all rectangles in the final result are composed of boundary edges and/or parts of chords.*

Proof By induction on the number of rectangles in the final result.
Induction Base: The expansion of the first rectangle examined in Step 2 of the heuristic stops only when we meet the boundary of the region on all four sides. Some of the sides of this expanded rectangle may be formed fully of a boundary edge (either all or part of a boundary edge) of the region while others may be only partially formed by boundary edges. The latter kind of sides would have met one or more concave vertices (Figure 5). The remainder of each such side must be formed by a horizontal or vertical chord drawn from a concave vertex that this particular side meets.

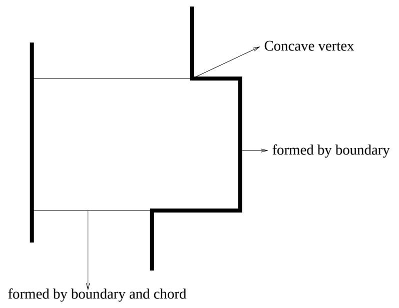


Figure 5: First rectangle formed by Heuristic 1

Induction Hypothesis: Assume that the first k expanded/finalized rectangles have their sides composed of boundary edges and/or chords.
Induction Step: The expansion of the $(k+1)$st rectangle being expanded stops only when the expanded

rectangle meets the boundary of the region or that of one or more of the previously expanded k rectangles. We note that:

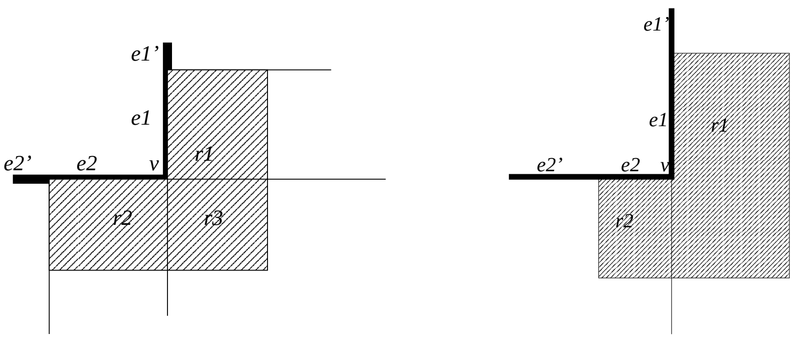
(a) Each side that meets the boundary of the region either is composed fully of a boundary edge or is composed of boundary edges and chords drawn from the concave vertices that the side meets.

(b) From the induction hypothesis, it follows that each side that meets the boundary of a finalized rectangle but not the boundary of the region is composed of parts of the chords or chord parts that form this boundary of the finalized rectangle that is met.

From (a) and (b) it is clear that all sides of the $(k+1)$st finalized rectangle are composed of the boundary edges of the region and/or some portion of chords of the region. ■

Lemma 2 *Every concave vertex in a non-degenerate region is a vertex of exactly one finalized rectangle.*

Proof First we prove that every concave vertex is a vertex of at least one finalized rectangle. Since all of the non-degenerate region is covered by finalized rectangles, the portions $e1$ and $e2$ of the boundary edges closest to any concave vertex v must lie on the boundary of two different finalized rectangles $r1$ and $r2$. Figure 6 shows two possibilities for $r1$ and $r2$.



(a) 3 rectangles at v (b) 2 rectangles

Figure 6: Finalized rectangles at a concave vertex

If $r1$ terminates at v (as in Figure 6(a)), then v is a vertex of $r1$. If $r1$ extends beyond v (as in Figure 6(b)), then v is a vertex of $r2$. In either case, v is a vertex of a finalized rectangle.

Next, we provide a proof by contradiction that each concave vertex can be a vertex of at most one finalized rectangle. Suppose that there is a concave vertex v that is a vertex of more than one finalized rectangle. Since a concave vertex makes an interior angle of 270 degrees and each vertex of a finalized rectangle makes an interior angle of 90 degrees, there must be exactly three finalized rectangles, $r1$, $r2$, and $r3$, that have v as a vertex (Figure 7).

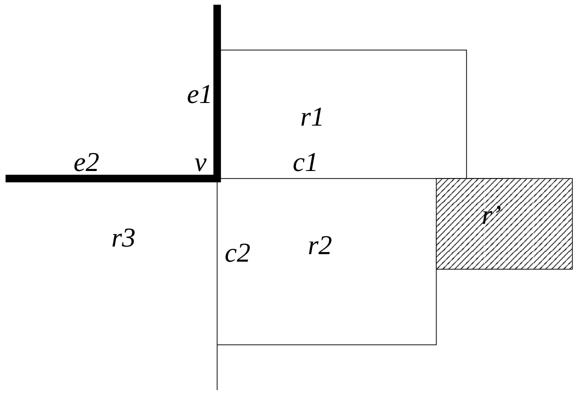


Figure 7: Vertex with 3 finalized rectangles

Without loss of generality, assume that $r1$ was the first of these three rectangles to be finalized. For the downward expansion of $r1$ to terminate at v, side $c1$ must meet a horizontal boundary edge or a horizontal side of an already finalized rectangle (r' in Figure 7). Both possibilities require another concave vertex v' such that (v, v') is a degenerate chord (Lemma 1). Since the region is non-degenerate, there can be no degenerate chord. ■

Corollary 1 *When Heuristic 1 is used to partition a non-degenerate region, there is exactly one chord drawn*[2] *from each concave vertex.*

Proof If two chords $c1$ and $c2$ are drawn from a concave vertex v then v is a vertex of 3 rectangles namely $r1$, $r2$, and $r3$ (Figure 7). ■

Lemma 3 *Suppose that a non-degenerate region is partitioned using Heuristic 1. In the final result, the chords drawn from the concave vertices form only 'T'-intersections at their other terminating end.*

Proof Consider the chord c drawn from the concave vertex v. There are two possiblities for where c ends. (1) c ends at a boundary of the non-degenerate region. In this case c and this boundary must form a 'T' (otherwise, c is a degenerate chord). (2) c ends at a side of a finalized rectangle r. c cannot be a side (or part of a side) of r nor can it end at a vertex of r (if it were/did, c wouldn't end at a side of r). Therefore, c forms a 'T' intersection with the side of r that it meets. ■

[2]The finalized rectangles may be viewed as the result of drawing chords so as to divide the region into rectangles.

Corollary 2 *For non-degenerate regions, the terminating end of each chord forms a vertex of exactly two finalized rectangles.*

Proof Since the terminating end v of chord c forms a 'T'-intersection and since c lies within the region, there must be a finalized rectangle on either side of c that has v as a vertex. (Figure 8). ■

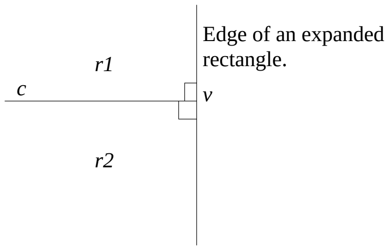


Figure 8: T-intersection formed by a chord

Theorem 1 *For non-degenerate rectilinear regions, the number of finalized rectangles is minimum.*

Proof From Lemma 1, it follows that each vertex of a finalized rectangle is either an original vertex of the region or is an end point of a chord. A convex vertex must be a vertex of exactly one finalized rectangle (convex vertices cannot be shared by 2 or more rectangles), a concave vertex is also a vertex of exactly one finalized rectangle (Lemma 2), and the terminating end of each chord is a vertex of exactly two rectangles (Corollary 2).

Therefore the total number of vertices in the finalized rectangles is number of convex vertices + number of concave vertices $+2*$number of chords $= n + 2*$number of concave vertices, where n is the number of vertices in the region (Lemma 2). Therefore, the number of finalized rectangles $= (n + 2*$number of concave vertices$)/4$. Ohtsuki [6] has shown that this is the minimum number of rectangles needed to partition a non-degenerate rectilinear region. ■

Corollary 3 *Heuristic 1 produces optimal partitions for 2D non-degenerate phantoms.*

Theorem 2 *For degenerate rectilinear regions the number of finalized rectangles may exceed the minimum number of rectangles by as much as the number of independent degenerate chords that can be drawn in the region.*

Proof The maximum number of finalized rectangles is $(n + 2*$number of concave vertices$)/4 = n/2 + h - 1$, where n is the number of vertices and h is the number of holes. This corresponds to the case when a distinct chord is drawn from each concave vertex. Ohtsuki [6] has shown that the minimum number of rectangles needed to partition a rectilinear region is $n/2 + h - d - 1$, where d is the maximum number of independent degenerate chords. Hence the number of finalized rectangles is at most d more than the minimum possible. ■

As an example, consider the degenerate rectilinear region shown in Figure 9. For this region, $n = 20, h = 2$, and $d = 6$. So, the minimum number of rectangles in any partitioning is $20/2 + 2 - 1 - 6 = 5$. Heuristic 1 could give the partitioning of Figure 9, which has $n/2 + h - 1 = 20/2 + 2 - 1 = 11$. This is because Heuristic 1 may draw no degenerate chords. Therefore, the ratio of number of rectangles given by Heuristic 1 to the minimum number of rectangles is $11/5 = 2.2$. By increasing the number of holes in the example of Figure 9 from 2 to i (simply add more vertical slabs), this ratio becomes $(5 + 3i)/(3 + i)$, which approaches 3 as i becomes large.

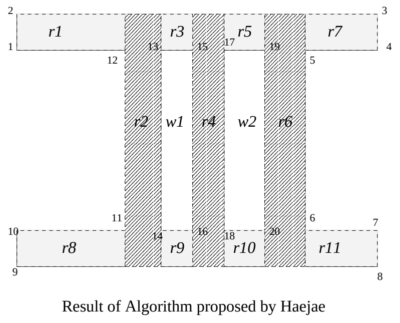


Figure 9: Finalized rectangles from Heuristic 1

For hole-free regions, the number of concave vertices is $n/2 - 2$ [7]. Since, in the worst case, d = number of concave vertices/2, the minimum number of rectangles in the partitioning of a hole-free rectilinear reagion may be as small as $(n/2 + 0 - d - 1) = n/2 - 1 - (n/2 - 2)/2$ while the number of finalized rectangles could be as large as $n/2 + 0 - 1$. Therefore, the ratio of number of rectangles given by Heuristic 1 to the minimum number of rectangles is $(n/2 - 1)/(n/2 - 1 - (n/2 - 2)/2) = 2 - 4/n$. This ratio approaches 2 as n becomes large.

4 Analysis for 3D Phantoms (3DPPP)

In this section, we show that Heuristic 1 is non-optimal even for simple 3D phantoms. We also show that slicing partitions are optimal for simple 3D phantoms. Since, voxels of one component of the phantom cannot be in the same cuboid as voxels of a different component, it is sufficient to analyze Heuristic 1 for an instance of 3DRPP in which the interior voxels are to be partitioned into a minimum number of cuboids.

4.1 Definitions

Definition 8 **Surface**–*This refers to the boundary faces of the region.*

Definition 9 **Convex edge**–*An edge for which the 2 adjacent surfaces make an interior angle of 90 degrees.*

Concave edge–*An edge for which the 2 adjacent surfaces make an interior angle of 270 degrees.*

Totally concave vertex–*Vertex at which all incident edges are concave edges.*

Totally convex vertex–*Vertex at which all incident edges are convex edges.*

We use the notation $c-vertex$ to refer to a totally convex vertex and $k-vertex$ to refer to a vertex that is not totally convex.

Examples are shown in Figure 1(b).

Definition 10 **Simple region**–*A region that satisfies the following two conditions:*

1. *No surface that has a concave edge may be extended over one or more of its concave edges to reach another concave edge.*

2. *There is no surface with two or more concave edges with the property that two of these concave edges can be joined by a line segment in the $x-y$, $y-z$ or $z-x$ planes, that lies wholly outside the surface, except at the two end points.*

A **simple phantom** *is a phantom, all of whose components are simple regions.*

For an example of a simple region, see Figure 1(b). Each edge of the inner cuboid, which is hole, is concave. It isn't possible to draw a line between any two concave edges such that the line lies in the $x-y$, $y-z$ or $z-x$ planes and is not contained wholly within a surface, as each surface is rectangular.

Figure 10 shows an example non-simple region. The shaded area depicts a hole. *p3* is a boundary surface of the hole and *e13* and *e23* are concave edges. The line *l1* that joins *e13* and *e23* does not lie wholly in a surface.

Definition 11 **Degenerate concave edges**–*Two concave edges form a degenerate pair iff they are not the edges of the same surface and they satisfy the following conditions (Figure 11):*

1. *They are parallel to the same axis.*

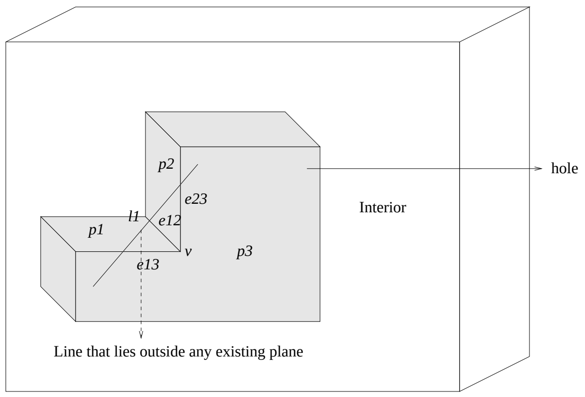


Figure 10: Non-simple region

2. *They can be joined by a line segment parallel to one of the axes, lying wholly outside any surface except at the two end points where the line segment joins the two edges.*

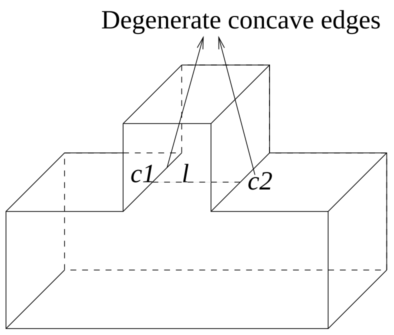
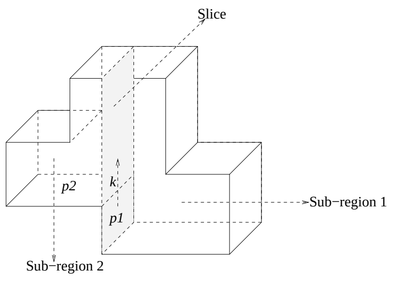


Figure 11: Degenerate concave edges

Definition 12 Slice–*A slice is an extension of a surface adjacent to a concave edge in a 3D region, such that it divides the 3D region into 2 or more sub-regions. For an extended surface to qualify as a slice, all its area should lie in the interior of the region (surface is considered as interior). All the edges of the slice lie wholly on surfaces of the region and/or, on perpendicular slices (Figures 12 and 14) .*

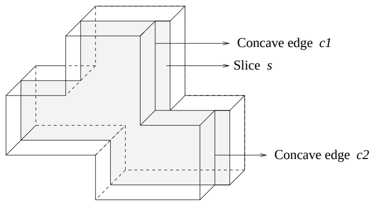
Definition 13 Slice segment–*A slice segment is a rectilinear extension of a surface adjacent to a concave edge in a 3D region. For an extended surface to qualify as a slice segment, all its area should lie in the interior of the region (surface is considered as interior) (Figures 13, 14, 15 and 16). Note that all slices are slice segments, but the converse is not true.*

Definition 14 Planes–*Apart from the usual meaning of the word, it is used to collectively refer to the slice segments and the surfaces in a 3D region at any particular time.*



Slice obtained by extending *p1*
adjacent to concave edge *k*

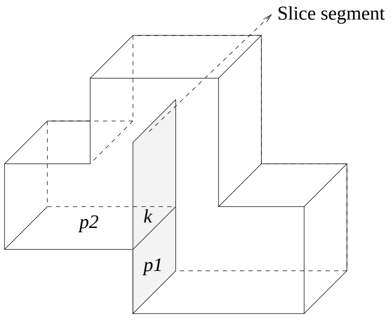
(a) Rectangular slice



Example of a slice through two concave edges in a non−simple region
This slice intersects several concave edges.

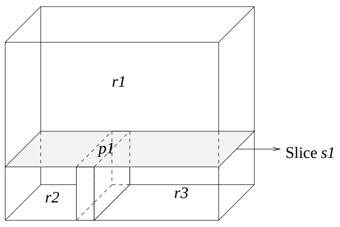
(b) Non-rectangular slice

Figure 12: Slices



Slice segment obtained by extending *p1*
adjacent to concave edge *k*

Figure 13: Slice segment



Example of a slice dividing region into 3 sub−regions

Figure 14: Horizontal slice

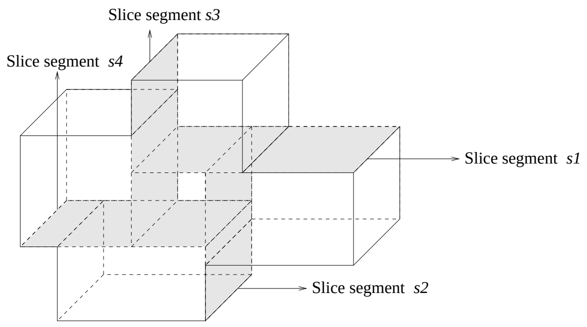


Figure 15: A partitioning obtained by slice segments

Definition 15 Slice segment partitioning (SSP)–*A partitioning of a rectilinear 3D region into cuboids, in which slice segments are formed for concave edges by extending a surface adjacent to each concave edge such that the extended surface cuts the 270 degree angle of the concave edge into a 90 degree angle and a 180 degree angle. The slice segments in the partitioned region satisfy the following conditions:*

1. *Each slice segment is bounded by a surface or some other slice segment along all its edges.*
2. *There is exactly one slice segment per concave edge, unless the concave edge is cut perpendicularly by some other slice segment.*
3. *If a concave edge is cut perpendicularly by a slice segment, then the two sub-concave edges obtained can be treated as independent concave edges and follow the same rule as described in condition 2.*

Note that condition 1 of Definition 10 and condition 3 of Definition 15 together imply that in every SSP of a simple region, there is exactly one slice segment per concave edge.

Definition 16 Slicing algorithm–*Any algorithm for 3DRPP that obtains an SSP by drawing slices for each concave edge in the region, in some sequence, till no concave edge remains. The resulting partitioning is called a* **slicing partition** *(SP).*

We distinguish between an edge of a partitioning and an edge of the 3D region that is to be partitioned. An *edge of a partitioning* is a maximally long line of the partitioning. For example, some of the edges in the SP of Figure 16 are $(v5, v6)$, $(v6, v1, v13)$, and $(v12, v7, v14)$. Note that $(v12, v7)$ is not an edge of this partitioning, because $(v12, v7)$ may be extended to the longer line $(v12, v7, v14)$ that is also a line of the partitioning. Every edge has exactly two end (or terminating) points and zero or more pass-through (intermediate) vertices. For example, the end points of the edge $(v5, v6)$ are $v5$ and $v6$. We say that

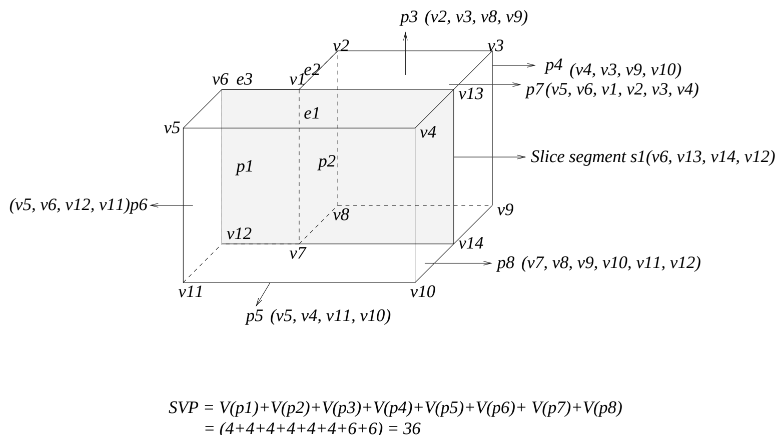


Figure 16: SP of a simple region

$(v5, v6)$ terminates at vertices $v5$ and $v6$. This edge has no pass-through vertices. The end points of the edge $(v12, v7, v14)$ are $v12$ and $v14$, and the edge has the single pass-through vertex $v7$.

Every edge of a partitioning is of one of the following types: (Figure 16):

1. **A new edge**–An edge such that no part of the edge existed in the original region. Example: $(v13, v14)$
2. **An extended edge**–An edge for which some part of the edge already existed in the region and the rest of it did not. Example: $(v6, v1, v13)$.
3. **An original edge**–The whole edge existed in the region. Example: $(v5, v11)$.

4.2 SSPs and Heuristic 1

Lemma 4 *Heuristic 1 is not a slicing algorithm.*

Proof To see this, examine Figure 17. This figure depicts a partitioning that could be generated by Heuristic 1. The cuboids are indexed 1 through 5 in the order they were created by the heuristic. The partitioning of Figure 17 cannot be obtained by a slicing algorithm, because in a slicing algorithm the slices need to be drawn in some sequence. Therefore, the first slice to be drawn would have all its edges on boundary surfaces. However, there is no extension of a surface in Figure 17 that meets this requirement. ■

Even though Heuristic 1 is not a slicing algorithm, it produces SSPs.

Lemma 5 *Heuristic 1 yields an SSP.*

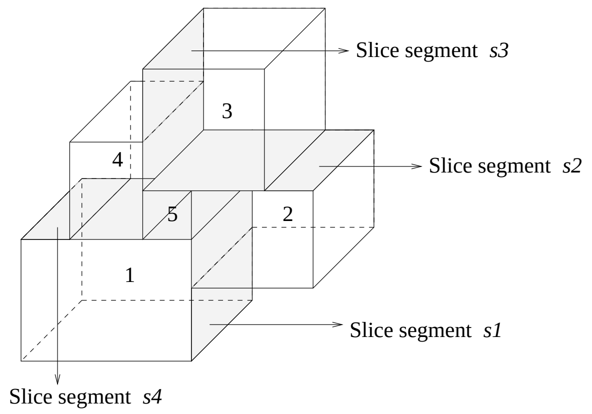


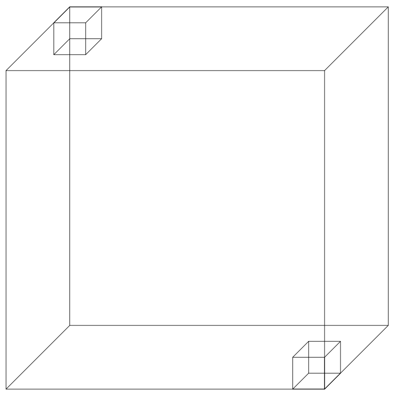
Figure 17: Partitioning obtained by Algorithm 1

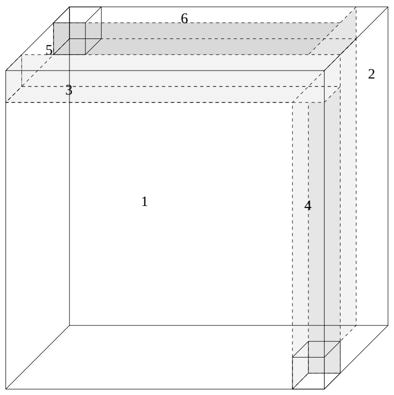
Proof It can be proved by induction that the faces of each cuboid in the final result of Algorithm 1, are composed of boundary surfaces of the original region or extensions of these boundary surfaces. Therefore, in the final result, all the cuboid boundaries are original surfaces and/or extension of some of the surfaces. The extension of a surface would be bounded by perpendicular boundary surfaces or extensions of some other surfaces, along all its edges. Therefore, any extended surface can be considered as a slice segment, and each slice segment satisfies all the conditions required for the partitioning to be called SSP. ■

Lemma 6 *Heuristic 1 may generate a non-optimal partitioning for a simple 3D region.*

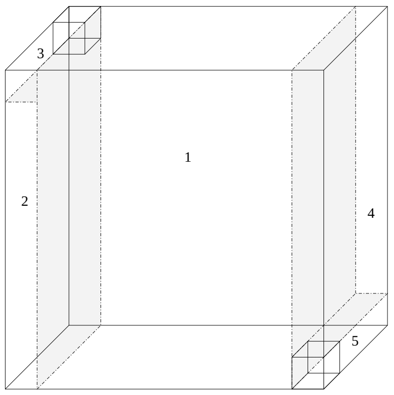
Proof Consider the simple 3D region of Figure 18(a). This region is a cuboid with two opposite corners removed. The removed corners are of different dimensions. Figure 18(b) shows the partitioning obtained by Heuristic 1. The 6 cuboids are numbered in the order in which they are generated by the heuristic. Notcie that the partitioning is an SSP but not an SP. Figure 18(c) shows a 5 cuboid SP for the simple 3D region of Figure 18(a). ■

It is easy to see that, for general 3D regions, the worst-case ratio of number of cuboids given by Heuristic 1 to the minimum number of cuboids is at least as large as the corresponding ratio for 2DRPP. Figure 19(a) shows an example 3D region for which this ratio approaches ∞ as the example is made larger. The shown non-simple region has 5 externally protruding horizontal slots on its right side and 5 externally protruding vertical slots on its left side. Assume that each horizontal slot is 1 voxel wide and one voxel high, and that each vertical slot is one voxel wide and one voxel deep. The maximum width of the region is 3 voxels. Note that the voxels are not drawn to scale. The middle voxels have been drawn wider then the left- and right-end voxels so as to avoid cluttering the figure. Figure 19(b)


(a) Simple region



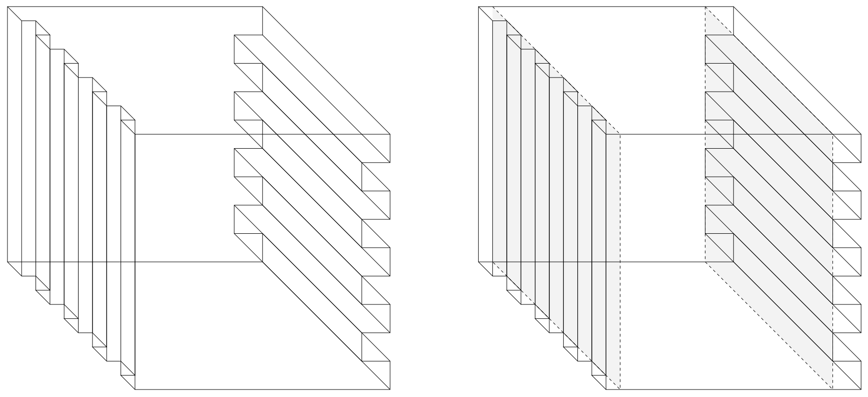
(b) SSP partitioning



(c) SP partitioning

Figure 18: SSP and SP partitionings of a simple 3D region

shows an optimal partitioning of the 3D region of Figure 19(a). This optimal partitioning has exactly 11 cuboids–5 vertical cuboids each of depth and width 1 and height 9, 5 horizontal cuboids of width and height 1 and depth 9, and 1 cuboid whose width is 1 and whose height and depth are 9. Figure 19(c) shows a partitioning that Heuristic 1 may produce. On the front face, we have 5 cuboids whose height and depth is 1 and whose width is 3. Between every two such cuboids on the front face is a cuboid whose height and depth is 1 and whose width is 2. Behind the front face of cuboids is a similar arrangement of cuboids whose width is one less than that of the cuboids on the front face. The pattern of 9 cuboids of height and depth 1 repeats to the back face, giving us a total of 81 cuboids. Extending this construction from 5 protruding slots on the left and right side, to n protruding slots on the left and right side, we get a non-simple 3D region for which the Heuristic 1 could produce a partitioning that has $(2n-1)^2$ cuboids while the optimal partitioning has $2n+1$ cuboids.


(a) Non-simple region (b) Optimal partitioning

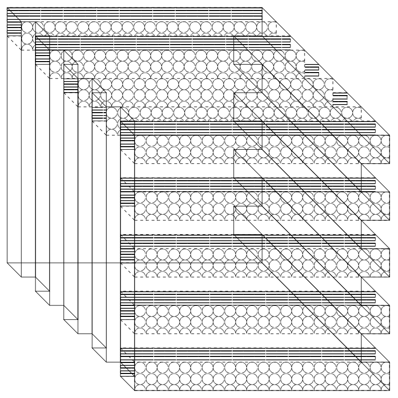

(c) Heuristic 1

Figure 19: Bad example for Heuristic 1

5 Slicing Partitions

Having established the non-optimality of Heuristic 1 for 3D simple regions, we turn our attention to slicing partitions. First, observe that slicing partitions may not be optimal for non-simple 3D regions. In fact, even though the optimal partitioning of Figure 19(b) is a slicing partition, other slicing partitions of the non-simple region of Figure 19(a) may not be optimal. For example, consider the slicing partition that results from making 8 horizontal slices that extend the 8 concave edges on the right side of the region (Figure 20). We are now left with 9 non-simple regions. In each of these, we make 8 slices parallel to the front face and extending the 8 vertical concave edges. Each of the 9 simple regions decomposes into 9 cuboids, for a total of 81 cuboids. By generalizing the region of Figure 19(a) to one that has n slots on the left and right ends, we get a 3D non-simple region for which an SP may have $(2n-1)^2$ cuboids, whereas the optimal partitioning has only $2n+1$ cuboids.

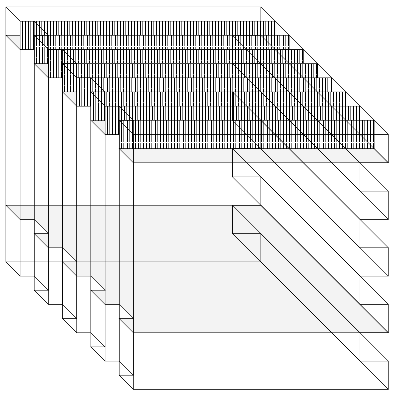


Figure 20: Possible slicing partition for Figure 19(a)

In the sequel, we first show that all slicing partitions of a 3D simple region yield the same number of cuboids. Note that this property does not hold for SSPs. This statement follows from the example of Figure 18 and observation that an SP is a special case of an SSP. Then we show that slicing partitions are optimal for simple regions.

Lemma 7 *In an SP of a simple region, every vertex that was originally present in the region is a vertex of exactly one cuboid.*

Proof An original vertex that is totally convex (i.e., a c-vertex) can be a vertex of only one cuboid regardless of which surfaces are chosen to form slice segments because a totally convex vertex makes an internal angle of 90 degrees.

Consider a vertex v that is not totally convex (i.e., at least one of the edges incident at v is concave, a k-vertex). In an SP, a slice segment is drawn through every concave edge. When a slice segment is drawn through an edge, the edge remains as an edge only on one side of the slice segment. On the other side, the surface that had the edge as one of its edges, is extended through the edge and thus the edge no longer exists. Therefore, v also exists only on the former side of the slice segment. Since a slice segment is formed for every concave edge that is incident to v, v remains a vertex only in one sub-region in an SP. Since v is a vertex in only one sub-region, it can be a vertex of only one cuboid (Figures 21 and 22).

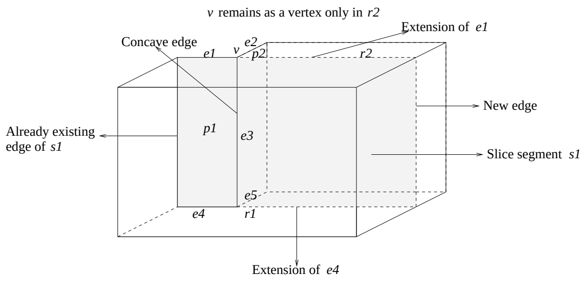


Figure 21: v is a vertex of cuboid $r2$

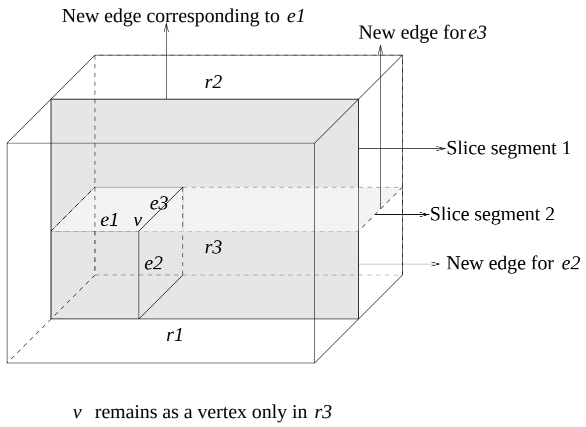


Figure 22: v is a vertex of cuboid $r3$

■

Lemma 8 *Every new vertex in an SP of a simple region is a vertex of exactly two cuboids.*

Proof A new vertex v is obtained when a slice segment meets a surface of the region or another slice segment, perpendicularly. In a simple region there is no concave edge meeting at v. So, the vertex v makes an interior angle of 90 degrees on either side of the slice segment. Since, the partitioned region

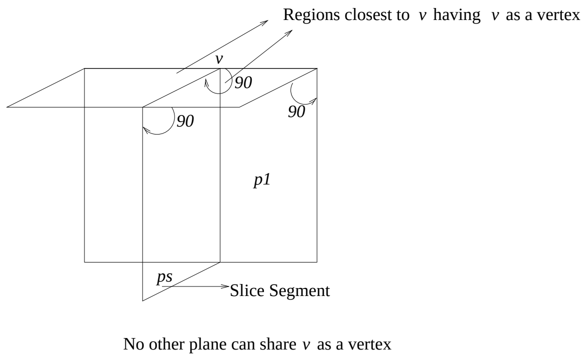

Regions closest to v having v as a vertex
v
90
90
90
p1
ps
Slice Segment
No other plane can share v as a vertex

Figure 23: New vertex v

is composed of disjoint cuboids, vertex v is a vertex of exactly two cuboids formed on either side of the slice segment containing the area closest to v (Figure 23).

■

Lemma 9 *In an SP of a simple region, every concave edge present in the original simple region corresponds to exactly one unique new edge.*

Proof As per the definition of an SP, when a slice segment is formed for a concave edge along one of the two possible planes, it meets the boundary of the original region or another slice segment, along all edges. For a concave edge lying on a slice segment, a new edge (as described in Definition 15) is obtained somewhere in the middle of a surface, or in the middle of a slice segment, opposite to the concave edge (Figure 21).

Since the region is simple, no other concave edge parallel to this new edge can lie on the slice segment. Also, no concave edge can be cut perpendicularly. Therefore, this new edge corresponds to exactly one unique concave edge, as there is only one slice segment per concave edge (as described in Definition 15).

■

Lemma 10 *Every new vertex of an SP of a simple region is the end point of exactly two edges of the SP, and every original vertex is the end point of either 2 or 3 edges.*

Proof From Lemma 9, it follows that every new vertex is a vertex of only the slice that created the vertex. The two edges of the slice that meet at this vertex terminate at this vertex. No other edge of the SP terminates at the new vertex (note that the new vertex has an additional edge that passes through it). So, every new vertex of an SP is the end point of exactly 2 edges of the SP.

The three edges that terminate at an original c-vertex of a simple 3D region also terminate at this vertex in every SP of the simple region. A k-vertex, however, has 2 terminating edges and 1 pass-through edge in the SP. ■

Definition 17 *An edge of a 3D region is a c-c edge if both its vertices are totally convex; it is a c-k edge if exactly one vertex is totally convex; and it is a k-k edge if it has no totally convex vertex.*

In the simple region of Figure 16 (ignore the slice $s1$), $(v5, v6)$ is a c-c edge, $(v1, v2)$ is a c-k edge ($v2$ is totally convex but $v1$ is not), and $(v1, v7)$ is a k-k edge.

Consider any partitioning of a 3D region. The *distinct-vertex count* (DVC) is the number of distinct vertices in the partitioning, and is equal to $N_{orig} + N_{new}$, where N_{orig} is the number of vertices originally in the 3D region and N_{new} is the number of new vertices. For the SP of Figure 16, $N_{orig} = 12$ and $N_{new} = 2$.

When exactly m edges of a partitioning have vertex v as their end point (i.e., when exactly m edges terminate at vertex v), we say that each of these m edges *contributes* $1/m$ to the vertex v. So, if the end points of an edge are vertices u and v, and if the edge contributes $1/m$ to u and $1/n$ to v, then the edge contributes $1/m + 1/n$ to the DVC for the partitioning. For the example of Figure 16, edge $(v1, v2)$ contributes 1/2 to vertex $v1$ (note that $v1$ is an end point of the edges $(v1, v2)$ and $(v1, v7)$) and 1/2 to vertex $v2$. Therefore, edge $(v1, v2)$ contributes $1/2 + 1/2 = 1$ to DVC. The edge $(v12, v7, v14)$ contributes 1/3 to vertex $v12$, 0 to vertex $v7$ ($v7$ is not an end point of this edge), and 1/2 to vertex $v14$. Therefore, this edge contributes $1/3 + 1/2 = 5/6$ to DVC.

Since every vertex of an SP is the end point of either 2 or 3 edges (Lemma 10), by adding together the contributions of all edges in an SP, we get the number of distinct vertices in the SP. An arbitrary partitioning of a simple region may, however, have vertices that are the end point of no edge. Therefore, the sum of the edge contributions for an arbitrary partitioning provides only a lower bound on the number of distinct vertices in the partitioning.

Lemma 11 *The contribution of an edge to the DVC of an SP as well as to that of an arbitrary partitioning are as summarized in Table 1.*

Proof First, note that in a 3D region, at most 3 edges may terminate at any vertex. Next, note that at most 2 edges may terminate at the end point of a new edge (this follows from the observation that the region interior lies to either side of the new edge). Therefore, a new edge contributes at least 1/2 +

Edge Type	SP	Arbitrary
New	1	≥ 1
c-c	2/3	2/3
c-k	5/6	$\geq 5/6$
k-k	1	≥ 1

Table 1: Edge contributions to DVC of an SP and of an arbitrary partitioning

$1/2 = 1$ to the DVC of a partitioning. Since the end points of a new edge are new vertices, Lemma 10 implies that, in an SP, each end point of a new edge contributes exactly 1/2 to the DVC. Therefore, every new edge of an SP contributes exactly 1 to the DVC of the SP.

Observe that the totally convex end (i.e., the c end) of an original edge is not extended in any partitioning and that 3 edges terminate at this end of the edge. Therefore, this end contributes 1/3 to the DVC. The k end of an original edge may or may not be extended in a partitioning (in Figure 16, $(v12, v7)$ is extended to $(v12, v7, v14)$, but $(v7, v8)$ is not extended). If the k end is extended, the edge terminates at a new vertex. From Lemma 10, we know that, in an SP, exactly two edges terminate at this new vertex. Further, in an arbitrary partitioning, at most 2 edges terminate at this new vertex. So, when the k end is extended, the k end contributes 1/2 to the DVC of an SP and at least 1/2 to the DVC of an arbitrary partitioning. If the k end is not extended in the partitioining, then at least one of the other original edges that terminates at this k end has to be extended so as to subdivide the 270 degree interior angle at this k end. Once again, we see that the k end contributes 1/2 to the DVC of an SP and at least 1/2 to that of an arbitrary partitioning. ■

Theorem 3 *The number of cuboids in every SP of a simple region is the same.*

Proof From Lemma 9, the number of new edges in every SP equals the number of concave edges in the simple region. Therefore, every SP has the same number of new edges, and so the contribution of these new edges to the DVC of every SP is the same. Since the number of c-c, c-k, and k-k edges are a property of the simple region and not of the partitioning, their contribution to the DVC is also the same for every SP. Finally, since the sum of the edge end point contributions equals the number of distinct vertices in an SP, every SP has the same DVC.

The number of new vertices in a partitioning, N_{new}, equals $DVC - N_{orig}$. From Lemma 8, every new vertex of an SP is the vertex of exactly 2 cuboids and from Lemma 7, every original vertex is the vertex of exactly 1 cuboid of the SP. Therefore, the number, N_{eff}, of effective vertices (i.e., weighted sum of number of vertices, each vertex is weighted by the number of cuboids it is a vertex of) in an SP equals $N_{orig} + 2 * N_{new} = 2 * DVC - N_{orig}$ is the same in every SP. Since the number of cuboids in a partitioning is $N_{eff}/8$, every SP has the same number of cuboids. ■

Let $\#concave$ be the number of concave edges in a simple 3D region and let $\#cc$ be the number of c-c edges. $\#ck$ and $\#kk$ are similarly defined. From the proof of Theorem 3, it follows that the number of cuboids in an SP of an n vertex 3D simple region is given by

$$[2 * (\#cc * (2/3) + \#ck * (5/6) + \#kk + \#concave) - n]/8$$

Theorem 4 *Every slicing partition of a simple region is optimal.*

Proof We need to show that the number of cuboids in an SP of a simple region is $\leq$ the number of cuboids in any other partitioining of a simple region. We do this in two parts. First, we show that the DVC, $DVC(SP)$, for an SP is $\leq$ the DVC, $DVC(arb)$, for any arbitrary partition. Next, we show that $DVC(SP) \leq DVC(arb)$ implies that the number of cuboids in the SP is $\leq$ the number of cuboids in the arbitrary partitioning.

For every concave edge in a simple region there is at least one distinct corresponding new edge in the partitioning. From Lemma 9, the number of new edges in an SP equals the number of concave edges. Therefore, the total contribution to $DVC(SP)$ from new edges equals the number of concave edges, whereas the contribution to $DVC(arb)$ is at least this much. Since the number of c-c, c-k, and k-k edges are a property of the simple region and not of the partitioning, their contribution to $DVC(SP)$ is no more than their contribution to $DVC(arb)$. Finally, the sum of the edge end point contributions equals $DVC(SP)$, but is $\leq DVC(arb)$ (because, some vertices of an arbitrary partition may not be the end point of an edge). Therefore, $DVC(SP) \leq DVC(arb)$.

For the second part of the proof, we see that $N_{new}(SP) = DCV(SP) - N_{orig}$ and $N_{new}(arb) = DVC(arb) - N_{orig}$. Therefore, $N_{new}(SP) \leq N_{new}(arb)$. Observe that every new vertex of an arbitrary partitioning is a vertex of at least 2 cuboids and every original vertex is the vertex of at least 1 cuboid. From this observation and the proof of Theorem 3, $N_{eff}(SP) = N_{orig} + 2 * N_{new}(SP) \leq N_{orig} + 2 * N_{new}(arb) \leq N_{eff}(arb)$. Since the number of cuboids in SP is $N_{eff}(SP)/8$ and that in an arbitrary partitioning is $N_{eff}(arb)/8$, the theorem follows. ■

6 New Heuristic

We may improve upon Heuristic 1 by developing a heuristic that favors the use of degenerate slice segments. Heuristic 2 is a simple heuristic that does this.

Heuristic 2

1 *From the given phantom of voxels of different densities choose one particular density and treat all the voxels of that density as interior voxels and the remaining voxels as exterior voxels. The interior voxels define one or more regions.*

2 *Form a list,* **concave***, of the concave edges in the regions defined in Step 1.*

3 *Pick an edge e ∈* ***concave****. There are two possible maximal slice segments (a slice segment is maximal if it cannot be extended further while retaining the slice properties) that include e. From these two segments, select and draw the slice segment that includes the larger number of concave edges.*

4 *Delete all concave edges (including e) that are included in the drawn slice segment from* ***concave****. Edges in* ***concave*** *that are cut by the slice segment are replaced by two concave edges that represent the two edge segments resulting from the cut.*

5 *Repeat Steps 3 and 4 until* ***concave*** *becomes empty.*

6 *Repeat Steps 1 through 5 for each of the different densities in the given phantom.*

We illustrate the working of our new heuristic using the 3D phantom of Figure 2(a). Without loss of generality, we assume that the new heursitic first partitions the region whose density is d1. Figure 24(a) depicts a possible slice that passes through 2 concave edges. However, this slice will not be choosen by the new heuristic, because the slice through both edges $e2$ and $e6$ passes through 4 concave edges. Note that in this phantom, regardless of which concave edge we begin with, for the region of density d1 we will always end up choosing a horizontal slice thus giving only 5 cuboids for density d1 and 2 cuboids always for density d2. Incidentally, this is also the optimal parttioning for this phantom. Figure 24(b) shows the partitioning obtained by the new heuristic.

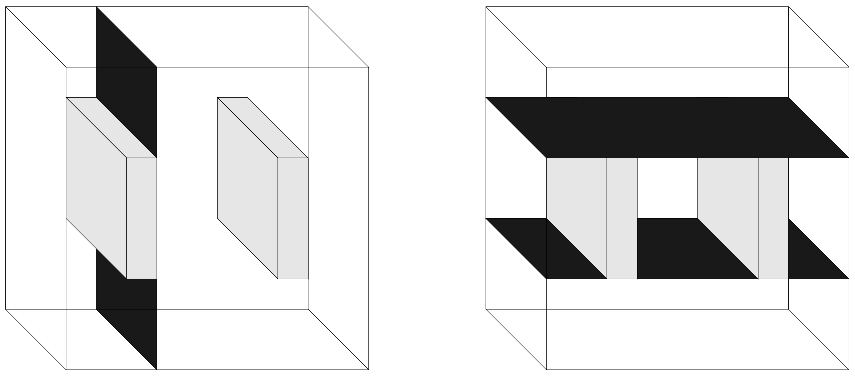

(a) A slice (b) Heuristic 2 partitioning

Figure 24: Example for Heuristic 2

7 Experimental Results

Heuristic 2 was implemented in C++ and benchmarked against Heuristic 1. Since neither of the heuristics was implemented with run-time efficiency in mind, we compare only the number of cuboids generated by each. For benchmarking, we used the $81 \times 81 \times 80$ voxel CT (computer tomography)-scan phantom used in [3] as well as several randomly generated phantoms.

The CT-scan phantom was first preprocessed, as in [3], so as to reduce the number of densities to 3. This was done using a standard binning/thresholding process as would be used in a real treatment planning system [3]. Heuristic 1 partitioned the 3-density CT-scan phantom into 10087 homogeneous cuboids, while Heuristic 2 partitioned the phantom into 3387 homogeneous cuboids, achieving a reduction in the number of cuboids by approximately 65%!

The randomly generated phantoms were created by initializing the density of each voxel to 1. Then, the density of a randomly selected $x\%$ of the voxels was changed to 2. Table 2 gives the characteristics of the random phantoms as well as the number of cuboids in the partitionings generated by the two heuristics. Heuristic 2 consistently generated at least 70% fewer cuboids.

8 Conclusion

We have shown that the phantom partitioning heuristic of [3] obtains optimal partitions for non-degenerate 2D but not for simple 3D phantoms. Slicing algorithms, on the other hand, generate optimal partitionings of simple 3D phantoms but not of general 3D phantoms. We have also proposed a simple

Phantom size	% density 2 voxels	#cuboids Heuristic 1	#cuboids Heuristic 2	% decrease
$30 \times 30 \times 30$	60	11546	2850	75
$30 \times 30 \times 30$	60	11479	2832	75
$30 \times 30 \times 30$	60	11439	2889	75
$30 \times 30 \times 30$	60	11518	2821	75
$30 \times 30 \times 30$	60	11518	2854	75
$30 \times 30 \times 30$	70	10624	2891	72
$30 \times 30 \times 30$	70	10544	2845	73
$30 \times 30 \times 30$	80	8934	2689	70
$40 \times 40 \times 40$	30	25132	6713	73
$40 \times 40 \times 40$	40	27216	6469	76
$40 \times 40 \times 40$	50	27778	6377	77
$50 \times 50 \times 50$	30	48998	12869	74
$50 \times 50 \times 50$	40	53140	12282	77

Table 2: Experimental results

heuristic for non-simple 3D phantoms. Experiments conducted using both randomly generated phantoms as well as a CT-scan phantom indicate that our heuristic generates far fewer cuboids than does the heuristic of [3].

References

[1] Victor J. Dielissen and Anne Kaldewaij. Rectangular partition is polynomial in two dimensions but NP-complete in three. *Information Processing Letters*, 38(1):1–6, April 1991.

[2] H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM J. Computing*, 15(2):478–494, May 1986.

[3] Haejae Jung. Algorithms for external beam dose computation. PhD dissertation, University of Florida, 2000. Chapter 4.

[4] Zuofeng Li and Jeffrey F. Williamson. Volume-based geometric modeling for radiation transport calculations. *Medical Physics*, 19(3):667–677, May/June 1992.

[5] Surendra Nahar and Sartaj Sahni. Fast algorithm for polygon decomposition. *IEEE Transactions on Computer-Aided-Design*, 7(4):473–483, April 1988.

[6] Tatsuo Ohtsuki. Minimum dissection of rectilinear regions. In *International Symposium on Circuits and Systems*, 1982. Pages 1210-1213.

[7] San-yuan Wu and Sartaj Sahni. Covering rectilinear polygons by rectangles. *IEEE Transactions on Computer-Aided-Design*, 9(4):377–388, April 1990.