JOURNAL OF SPATIAL INFORMATION SCIENCE
Number 21 (2020), pp. 137–174 doi:10.5311/JOSIS.2020.21.582

RESEARCH ARTICLE

Modeling orebody structures: Block merging algorithms and block model spatial restructuring strategies given mesh surfaces of geological boundaries

Raymond Leung

Australian Centre for Field Robotics, The University of Sydney, Australia

Received: October 13, 2019; returned: March 4, 2020; revised: June 9, 2020; accepted: August 12, 2020.

Abstract: This paper describes a framework for capturing geological structures in a 3D block model and improving its spatial fidelity, including the correction of stratigraphic, mineralization, and other types of boundaries, given new mesh surfaces. Using surfaces that represent geological boundaries, the objectives are to identify areas where refinement is needed, increase spatial resolution to minimize surface approximation error, reduce redundancy to increase the compactness of the model and identify the geological domain on a block-by-block basis. These objectives are fulfilled by four system components which perform block-surface overlap detection, spatial structure decomposition, sub-blocks consolidation, and block tagging, respectively. The main contributions are a coordinate-ascent merging algorithm and a flexible architecture for updating the spatial structure of a block model when given multiple surfaces, which emphasizes the ability to selectively retain or modify previously assigned block labels. The techniques employed include block-surface intersection analysis based on the separable axis theorem and ray-tracing for establishing the location of blocks relative to surfaces. To demonstrate the robustness and applicability of the proposed block merging strategy in a more narrow setting, it is used to reduce block fragmentation in an existing model where surfaces are not given and the minimum block size is fixed. To obtain further insight, a systematic comparison with octree subblocking subsequently illustrates the inherent constraints of dyadic hierarchical decomposition and the importance of inter-scale merging. The results show the proposed method produces merged blocks with less extreme aspect ratios and is highly amenable to parallel processing. The overall framework is applicable to orebody modeling given geological boundaries, and 3D segmentation more generally, where there is a need to delineate spatial regions using mesh surfaces within a block model.

Keywords: block merging algorithms, block model structure, spatial restructuring, mesh surfaces, subsurface modeling, geological structures, boundary correction, domain identification, iterative refinement, geospatial information system

© by the author(s) Licensed under Creative Commons Attribution 3.0 License

1 Introduction

This article considers the spatial interaction of triangle-mesh surfaces with a block model as defined by Poniewierski [24]. For this study, a block model may be conceptualized as a collection of rectangular prisms that span a modeled region in 3D space. Using a two-tier description, the block models of interest are formed initially by a set of non-overlapping 'parent' blocks; these blocks have identical dimensions and are evenly spaced so they form a regular 3D lattice. Furthermore, a subset of the parent blocks—particularly those that intersect with a surface—are decomposed into smaller cuboids (often referred as 'children' or sub-blocks) with the objective of preserving surface curvature subject to a minimum block size constraint. The sub-blocking problem represents a major theme in this paper. A distinguishing feature is that this problem is approached from a merging (bottom-up) perspective which offers opportunities for sub-block consolidation to minimize over-splitting, an issue often neglected in top-down approaches that focus exclusively on making block splitting decisions. The proposed framework allows new block models to be generated within a cell-based system using region partitioning surfaces. This comprises a spatial restructuring strategy that also allows iterative refinement of an existing block model given newer surfaces. At its core is a block merging algorithm that increases block model compaction and reduces spatial fragmentation due to subblocking. As motivation, we describe how this framework is deployed in a mine geology modeling system to illuminate key aspects of the proposal and illustrate what purpose they serve.

In mining, 3D geological models are used in resource assessment to characterize the spatial distribution of minerals in ore deposits [17]. A block model description of the geochemical composition is often created by fusing various sources of information from drilling campaigns, these include for instance: assay analysis, material, or geophysical logging, and alignment of stratigraphic units from geologic maps during the exploration phase. Due to the sparseness of these samples, the inherent resolution of these preliminary models are typically low. As the exploitation phase commences, denser samples may be taken strategically to develop a deeper understanding about the geology of viable ore deposits. This knowledge can assist miners with planning and various decision making processes [13], for instance, to prioritize areas of excavation, to develop a mining schedule [7], to optimize the quality of an ore blend in a production plant. Of particular relevance to spatial modeling is that wireframe surfaces can be generated by geo-modeling software [28] [20] [12], or via kriging [9], probabilistic boundary estimation [3], boundary propagation (differential geometry) [14], and other inference techniques [31] to minimize the uncertainty of interpolation at locations where data were previously unavailable. For instance, triangle meshes may be created by applying the marching cubes algorithm [22] to Gaussian process implicit surfaces [8]. These boundary updates provide an opportunity to refine existing block models and remove discrepancies with respect to verified boundaries. The objective is to maximize the model's fidelity by increasing both accuracy and precision subject to some spatial constraints. The desired outcomes are improved localization, reduced quantization errors and less spatial fragmentation. In other words, the boundary blocks in the block model should accurately reflect the location of boundaries between geological domains; smaller blocks should be used to capture the curvature of regions near boundaries to minimize the surface approximation error; the model should provide a compact representation and have a low block count to limit spatial fragmentation.

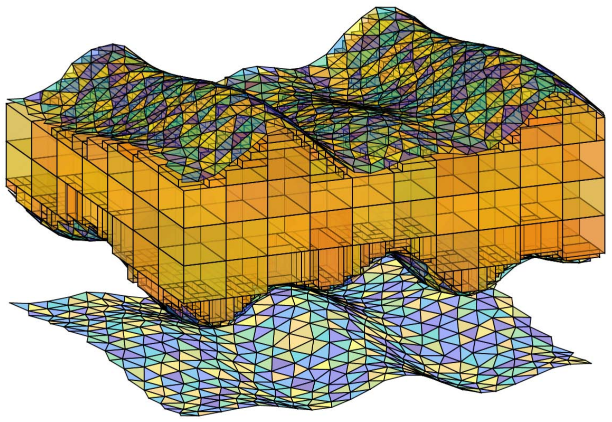


Figure 1: Essence of block model spatial restructuring given mesh surfaces. A total of three surfaces are involved. For clarity, only the orange blocks situated between the top and middle surfaces are shown.

Figure 1 provides a visual summary of the primary objective. A key feature of spatial restructuring is that blocks are divided as necessary to adapt the block model to the curvature of the given surfaces. This process, known as *sub-blocking*, is commonly performed in a top-down recursive manner which prioritizes splitting ahead of block consolidation. In some implementations, block consolidation is omitted altogether; this usually results in a highly fragmented and inefficient block representation. In this paper, surface-intersecting blocks are decomposed down to some minimum block size, then hierarchical block merging is performed in a bottom-up manner to consolidate the sub-blocks. In the ensuing sections, a framework for modifying the spatial structure of a block model using triangular mesh surfaces is first presented, the techniques underpinning each subsystem are described. Subsequently, we devote our attention to the block merging component, the algorithm is extended to support different forms of merging constraint. The proposed methods are applicable to orebody modeling given surfaces of mineralization or stratigraphic boundaries—see scenarios illustrated by [33], [10], [18], [4]; and general purpose 3D block-based modeling given other types of delineation.

1.1 Definition of a surface

In this paper, the term ‘surface’ encompasses both 2.5D and 3D surfaces. The former refers to open surfaces or warped 2D planes; these generally include mineralization, hydration, and stratigraphic surfaces in a mining context. The latter refers to closed surfaces that envelop a volume in 3D space. Examples include compact 2-manifolds that are topologically equivalent to a sphere. A simply-connected polyhedron surface would satisfy this requirement and have an Euler-Poincaré characteristic of 2. Closed surfaces may represent regions with local enrichment in an ore deposit or pockets with high level of contaminants.

1.2 Justification for a block-based approach

In our envisaged application, the resultant block structure facilitates volumetric estimation of geochemical or geometallurgical properties within a mine. A block-based representation provides spatial localization while block labelling provides differentiation between geological domains. These properties are important for two reasons. First, it enables spatially varying signals and non-stationary geostatistics to be learned and captured at an appropriate granularity for mining. Individual blocks can be populated with local estimates of the geochemistry, material type composition or other physical attributes. Second, it prevents sample averaging from being applied across discontinuities (different domains) which can lead to incorrect probabilistic inferences and skewed predictions if applied unknowingly across a boundary with a steep attribute gradient. Although polyhedral, topological, and surface-oriented models offer some interesting alternatives, block models have been well established and widely deployed within the resource industry. Introducing a radically different approach will likely cause significant disruption and require changes, acceptance and adaptation by various stakeholders. The proposed block model restructuring strategy is designed to work seamlessly with existing models. It makes changes which are transparent and compatible with subsequent processes such as mining exacavation which also operates on a block scale.

2 Framework for block model spatial restructuring

This section develops a framework for altering the spatial structure of a block model to reconcile with the shape of the supplied surfaces as depicted in Figure 1. The input block model consists of non-overlapping blocks of varying sizes (with uniform 3D space partitioning being a special case) and the surfaces, typically produced by boundary modeling techniques, represent the interface between different geological domains. The triangular mesh surfaces, together with the initial block model and block tagging instructions constitute the entire input. The tagging instructions simply assign to each block a label which classifies its location relative to each surface. The framework may be described in terms of four components: block surface overlap detection, block structure decomposition, sub-blocks consolidation, and block tagging (domain identification) as shown in Figure 2.

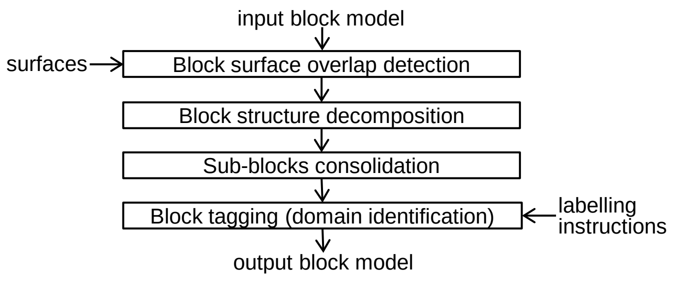

input block model
surfaces
Block surface overlap detection
Block structure decomposition
Sub-blocks consolidation
Block tagging (domain identification)
labelling instructions
output block model

Figure 2: Components in the block model spatial restructuring framework.

2.1 Block surface overlap detection

The goal is to identify blocks in the input model which intersect with the given surface(s). These represent areas where model refinement is needed in order to minimize the surface approximation error. To establish a sense of scale, the input blocks (also called "parent blocks") typically measure $25 \times 25 \times 5$m in an axis-aligned local frame, "axis-aligned" means the edges of each block are parallel to one of the (x, y, or z) axes.

2.2 Block structure decomposition

Block decomposition is performed on surface-intersecting parent blocks to improve spatial localization. This process divides each block into smaller blocks of some minimum size (e.g., $6.25 \times 6.25 \times 1.25$m). These sub-blocks are disjoint and together, they span the whole parent block. Geometry tests are applied to determine which sub-block inside each surface-intersecting parent block actually intersects with a given surface. This process places blocks into one of three categories: (a) parent blocks that never intersect with any surface; or else sub-blocks inside surface-intersecting parent blocks that (b) intersect with a surface or (c) do not intersect with any surface.

2.3 Sub-blocks consolidation

This component coalesces sub-blocks to produce larger blocks in order to prevent over-segmentation. Recall that one of the objective is to minimize the total number of blocks in the output model. Therefore, if an array of $3 \times 4 \times 2$ sub-blocks could be merged into a perfect rectangular prism, it would be more efficient to represent them collectively by a single block with a new centroid and their combined dimensions.

In the event that the sub-blocks (within a surface-intersecting parent block) intersect with multiple different surfaces (say s_1 and s_2) then merging considerations will be applied separately to the sub-blocks intersecting with s_1 and s_2. It is important to observe that block consolidation is applied to both category "b" and category "c" sub-blocks as defined above. Also, merging of sub-blocks across parent block boundaries is not permitted. In Section 3.3, we describe in detail the proposed block merging approach which draws inspiration from the coordinate-ascent algorithm.

2.4 Block tagging (domain identification)

As a general principle, block tagging assigns abstract block labels to differentiate blocks located above and below a given surface. The objective is to support tagging with respect to multiple surfaces and two labelling policies. The first policy distinguishes surface-intersecting blocks from non-intersecting blocks. The second policy forces a binary decision and labels surface-intersecting blocks as strictly above or below a surface. The scheme also offers the flexibility of using a surface to limit the scope of an update, thereby leaving previously assigned labels intact above / below a surface. The processes described throughout (Section 2.1–2.4) are deterministic and applicable in an iterative setting. This completes our brief overview of the system.

3 Techniques

The techniques used in each component will now be further described. For maximum efficiency, the general set up assumes the modeled region aligns with the xyz axes. If the modeled deposit follows the inclination of a slope, it is assumed that an appropriate 3D rotation is applied to all spatial coordinates before subsequent techniques are applied. This has the effect of producing axis-aligned blocks in the modeling space that ultimately align with the principal orientation of the deposit in real space once the inverse transformation is applied at the conclusion of the process.

3.1 Detecting block surface intersection

The method we employed for determining if an axis-aligned rectangular prism intersects with a triangular patch from a surface is described by Akenine-Möller in [1]. This approach applies the Separating Axis Theorem (SAT) which states that two convex polyhedra, A and B, are disjoint if they can be separated along either an axis parallel to a normal of a face of either A or B, *or* along an orthogonal axis computed from the cross product of an edge from A with an edge from B. The technical details can be found in Appendix A.

In essence, block-surface intersection assessment consists of a series of "block versus triangle" comparisons where the triangles considered for each block are selected based on spatial proximity. The block-triangle overlap assessment operates on one basic principle: a "no intersection" decision is reached as soon as one of the tests returns FALSE. A block-triangle intersection is found only when all 13 tests return TRUE, when it failed to find any separation. These tests are used to identify the surface triangles (if any) that intersect with each parent block.

To maximize computation efficiency, each block is tested against a subset of triangles on each mesh surface rather than the entire mesh surface. A kD-tree accelerator (a variation of binary space partitioning tree) [23] is constructed using the 3D bounding boxes of each triangle. The subset includes only triangles whose bounding box overlaps with the block; only these candidates can intersect with the block. This pruning step limits the number of comparisons and speeds up computation considerably. The indices harvested here can subsequently reduce the test effort in the block structure decomposition stage.

3.2 Block structure decomposition

The key premise is to decompose surface-intersecting blocks into smaller blocks to improve spatial localization. The basic intuition is that the surface discretization error decreases as spatial resolution increases. Precision increases when smaller blocks are used to approximate the surface curvature where the blocks meet the surface.

Block structure decomposition entails the following. For each block (b) that intersects with the surface, divide it volumetrically into multiple sub-blocks using the specified minimum block dimensions ($\Delta_{x,\min}^{\text{block}}, \Delta_{y,\min}^{\text{block}}, \Delta_{z,\min}^{\text{block}}$). The main constraints are that sub-blocks cannot overlap and they must be completely contained by the parent block whose volume is the union of all associated sub-blocks.[1] Within each surface-intersecting parent block, we also identify all sub-blocks that intersect with a surface and which surface they intersect

[1]Although we typically require parent blocks ($\Delta_x^{\text{block}}[b], \Delta_y^{\text{block}}[b], \Delta_z^{\text{block}}[b]$) to be divisible by the minimum block dimensions, the method works fine even if fractional blocks emerge during the division, i.e., the last block

with. This is accomplished using the associative mapping [obtained during block-surface overlap detection] which limits the relevant surface triangles to a small subset for each surface-intersecting parent block. The relevant attributes captured in the output include a list of surface-intersecting parent blocks, and attributes for each sub-block: viz., its centroid, dimensions, parent block index, intersecting surface and position within the parent block.

3.3 Sub-blocks consolidation via coordinate-ascent

The consolidation component focuses on merging sub-blocks inside surface-intersecting parent blocks to reduce block fragmentation. Its basic objective is to minimize the block count, although other mining or geologically relevant criteria such as block aspect ratio can be optimized for. These options are considered later in Section 6.1 and Section 7.1. For now, these high resolution sub-blocks may themselves intersect or not intersect with any surface. This is indicated in the *BlockStructureDecomposition* result which serves as input. This component returns the *SubBlockConsolidation* result which describes the consolidated block structure. This encompasses all parent blocks processed, including those which do not intersect with any surface, as well as sub-blocks or super-blocks that constitute the surface-intersecting parent blocks.

The proposed merging algorithm is inspired by **coordinate-ascent optimisation** and may be summarized as follows.

- The algorithm is inspired by "coordinate ascent" where the search proceeds along successive coordinate directions in each iteration. The goal is to grow each block (a rectangular prism) from a single cell and find the maximum extent of spatial expansion, $\mathbf{k} = (k_x, k_y, k_z)$, without infringing other blocks or cells that belong to a different class.
- For merging purpose, each parent block is partitioned uniformly down to the minimum block size. The smallest unit (with minimum block size) is referred as a *cell*. Block dimensions are expressed in terms of the number of cells that span in the x, y, and z directions. Accordingly, if a block b with cell dimensions $\mathbf{k} = (k_x, k_y, k_z)$ is anchored at position $\mathbf{c}^{(b)} \in \mathbb{R}^3$, its bounding box would stretch from $\mathbf{c}^{(b)} - \frac{1}{2}\mathbf{\Delta}_{\text{min}}^{\text{block}}$ to $\mathbf{c}^{(b)} + (\mathbf{k} - \frac{1}{2})\mathbf{\Delta}_{\text{min}}^{\text{block}}$ for $k_x, k_y, k_z \in \mathbb{Z} \geq 1$.
- Merging states are managed using a binary occupancy map (boolean 3D array) with cell dimensions identical to the parent block. Given a block anchor position $\mathbf{c}^{(b)}$ with $\mathbf{k}$ initialized to $(1, 1, 1)$, expansion steps are considered in each direction $\boldsymbol{\delta} \in \mathbb{Z}^3$ which must alternate through the sequence $\boldsymbol{\delta}_0 = (1, 0, 0)$, $\boldsymbol{\delta}_1 = (0, 1, 0)$, and $\boldsymbol{\delta}_2 = (0, 0, 1)$.
- A step in the direction $\boldsymbol{\delta}$ is feasible if all of the cells within the bounding box ($\mathbf{c}^{(b)} - \frac{1}{2}\mathbf{\Delta}_{\text{min}}^{\text{block}}$, $\mathbf{c}^{(b)} + (\mathbf{k} + \boldsymbol{\delta} - \frac{1}{2})\mathbf{\Delta}_{\text{min}}^{\text{block}}$) are 1 (active). In this case, the spatial extent is updated via $\mathbf{k} \leftarrow \mathbf{k} + \boldsymbol{\delta}$. Each iteration steps through $\boldsymbol{\delta}_0$, $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$ in turn. This continues until no expansion is possible in any direction, at which point the centroid and dimensions of the merged block are computed and the corresponding cells in the occupancy map are set to 0 (marked as inactive).
- Sub-block merging terminates for a parent block when all cells in the occupancy map are set to 0.

toward the end is smaller than the minimum size; the ratios in each dimension $n_x^{\text{block}}, n_y^{\text{block}}, n_z^{\text{block}} \notin \mathbb{Z}$ can be non-integers. This essentially means there is no fundamental limit on the minimum spatial resolution.

- The volume of a consolidated block is effectively the outer product $[0, k_x) \otimes [0, k_y) \otimes [0, k_z)$ where k_x, k_y, k_z each represents some integer multiple of the minimum block size, $(\Delta^{\text{block}}_{x,\min}, \Delta^{\text{block}}_{y,\min}, \Delta^{\text{block}}_{z,\min})$, with respect to x, y, and z.
- Different coordinate directions are used cyclically during the procedure. At all times, it must ensure the expansion does not include alien blocks in the encompassing cube, e.g., an L-shape block within a $2 \times 2 \times 1$ cube is not allowed.

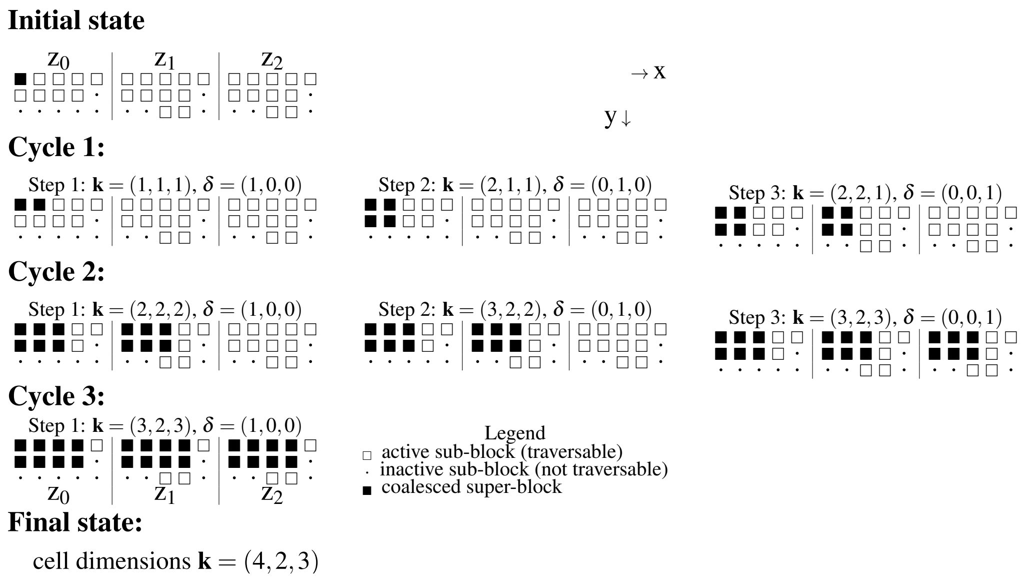

Initial state
Z0
Z1
Z2
→ x
y↓
Cycle 1:
Step 1: $\mathbf{k} = (1,1,1)$, $\delta = (1,0,0)$
Step 2: $\mathbf{k} = (2,1,1)$, $\delta = (0,1,0)$
Step 3: $\mathbf{k} = (2,2,1)$, $\delta = (0,0,1)$
Cycle 2:
Step 1: $\mathbf{k} = (2,2,2)$, $\delta = (1,0,0)$
Step 2: $\mathbf{k} = (3,2,2)$, $\delta = (0,1,0)$
Step 3: $\mathbf{k} = (3,2,3)$, $\delta = (0,0,1)$
Cycle 3:
Step 1: $\mathbf{k} = (3,2,3)$, $\delta = (1,0,0)$
Legend
□ active sub-block (traversable)
· inactive sub-block (not traversable)
■ coalesced super-block
Final state:
cell dimensions $\mathbf{k} = (4,2,3)$

Figure 3: Sub-blocks consolidation: illustration of the coordinate-ascent merging algorithm using occupancy map.

The algorithm is explained by way of an example in Figure 3. To illustrate the algorithm, let us refer to blocks that belong to the class under consideration as the **active** cells. In the present context, this refers to either surface-intersecting sub-blocks, $\mathcal{S}_{\text{intersect}}$, or the non-intersecting sub-blocks, $\mathcal{S}_{\text{non-intersect}}$, inside a parent block. The **inactive** cells (labelled 0) refer to the complementary set to the active cells (labelled 1).

In Figure 3, suppose the active sub-blocks (cells) refer to $\mathcal{S}_{\text{intersect}}$. The initial state shows the decomposition of a parent block in terms of three x/y cross-sections. There are $5 \times 3 \times 3$ sub-blocks (cells) within the parent block and 31 are considered "active". The white square cells all intersect with the same surface. It so happens the first cell encountered (with index $(i_Z \cdot n_Y + i_Y) \cdot n_X + i_X = 0$) is the first active block. The algorithm considers expansion along each axis (x, y, and z) in turn. In cycle 1, an expansion step in the x-direction is possible. The progressive expansion of the coalesced block is represented by black square cells. In cycle 2, further growth in the y-direction is not possible due to impediment by one or more "inactive" blocks (denoted by •), however, the x and z dimensions each allow one step expansion. In cycle 3, expansion continues in the x direction, resulting in the formation of a $4 \times 2 \times 3$ super-block. At this point, a coalesced block is extracted as no further growth in any direction is now possible. Subsequently, the cells which have just been merged are marked out-of-bounds. The process repeats itself, starting with the next active cell it encounters in the raster-scan order.

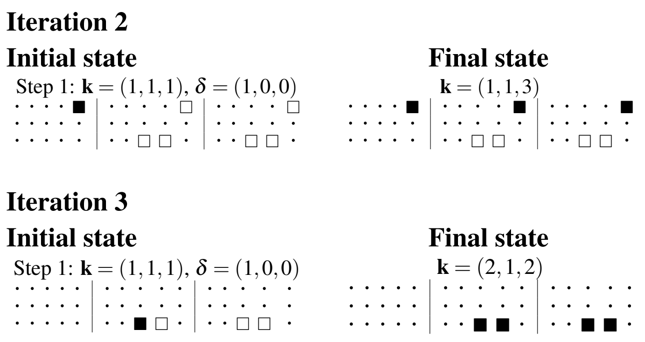


Figure 4: Sub-blocks consolidation (continued)—coalescing remaining active cells.

Once all the sub-blocks (cells) are coalesced within the surface-intersecting parent block, we are left with 3 merged blocks (colored in different shades of red in Figure 5). These have cell dimensions[2] (k_x, k_y, k_z) of $(4,2,3)$, $(1,1,3)$, and $(2,1,2)$ and relative centroids of $(\frac{4}{10}, \frac{2}{6}, \frac{3}{6})$, $(\frac{9}{10}, \frac{1}{6}, \frac{3}{6})$, and $(\frac{6}{10}, \frac{5}{6}, \frac{4}{6})$, respectively, with respect to the parent block's minimum vertex and parent block dimensions.

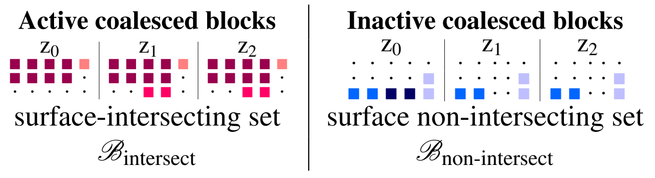

Active coalesced blocks
z_0
z_1
z_2
surface-intersecting set
$\mathscr{B}_{\text{intersect}}$
Inactive coalesced blocks
z_0
z_1
z_2
surface non-intersecting set
$\mathscr{B}_{\text{non-intersect}}$

Figure 5: Sub-blocks consolidation (continued)—consolidated blocks in the surface-intersecting set $\mathcal{S}_{\text{intersect}}$ and non-intersecting set $\mathcal{S}_{\text{non-intersect}}$.

- When **multiple surfaces** are considered, the coalesce function will next be applied to other sub-blocks (within the same parent block) that intersect with other surfaces. This seldom happens but it is possible if the surfaces are close.
- Finally, the coalesce function is **applied to inactive sub-blocks** (within the same parent block) which do not intersect with any surface. The result for inactive blocks (colored in different shades of blue) are shown in Figure 5.

To summarize, coalescing is applied only to surface-intersecting parent blocks which are subject to decomposition. The sub-blocks contained within may intersect with one or more surfaces, or not intersect with any surface at all. After consolidation, the output contains both the original and merged blocks. Based on the sub-block and surface intersection status, the consolidated blocks may be arranged into two separate sets, $\mathcal{B}_{\text{intersect}}$ and $\mathcal{B}_{\text{non-intersect}}$, as shown in Figure 5. Surface intersecting merged blocks 🟥🟥🟥 (within surface-intersecting parent blocks) are placed in $\mathcal{B}_{\text{intersect}}$. Non surface-intersecting merged blocks

[2]These represent multiplying factors relative to the minimum block size.

🟪🟦■ (within surface-intersecting parent blocks) are placed in $\mathcal{B}_{\text{non-intersect}}$. Non surface-intersecting parent blocks are also appended to this. The coordinate ascent algorithm provides a method for merging cells (sub-blocks of minimum size) within the confines of a parent block. Coalesced sub-blocks must share the same label and form a rectangular prism.

3.3.1 Practicalities

Since the cell division lines are identical for all parent blocks of the same size, to avoid having to compute the sub-block centroids and dimensions repeatedly, the cell dimensions and local coordinates of each sub-block's centroid (relative to the parent block's minimum vertex) are stored in a look-up table and indexed by parent block dimensions to speed up computation.

3.4 Block tagging: domain identification using surfaces

The primary objective is to label whether a block is located above or below a surface. A picture of this is shown in Figure 6. More generally, the surface is not always horizontal, so it makes more sense referring to the space on one side of the surface as *positive*, the other as *negative*, however this is defined.

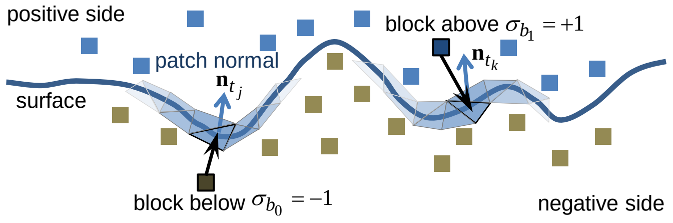

positive side
patch normal $\mathbf{n}_{t_j}$
block above $\sigma_{b_1} = +1$
$\mathbf{n}_{t_k}$
surface
block below $\sigma_{b_0} = -1$
negative side

Figure 6: Concepts relating to surfaces: surface normal (cross product) and block projected distance (inner product) associated with a triangular patch.

The main points to appreciate in Figure 6 are that

- Each triangular patch of the surface has a normal vector associated with it. For example, the normal $\mathbf{n}_{t_k}$ associated with triangle t_k points in the upward direction. This normal is computed by taking the cross product between two of its edges, for instance, $(\mathbf{v}_{k,1}-\mathbf{v}_{k,0})\times(\mathbf{v}_{k,2}-\mathbf{v}_{k,0})$. This arrow would point in the opposite direction (rotate by 180°) if the edge traversal direction is reversed; for instance, by swapping two vertices $\mathbf{v}_{k,1}$ and $\mathbf{v}_{k,2}$ in the triangle.
- To determine "which side of the surface" a block is on, it suffices to consider the triangular patch located closest to the block. After establishing the positive side as the space 'above' the surface, one can say block b_1 is located above the surface and has a positive sign ($\sigma_{b_1} = +1$) because the inner product $(\mathbf{c}_{t_k} - \mathbf{c}_{b_1}) \cdot \mathbf{n}_{t_k}$ is negative. Here, $\mathbf{n}_{t_k}$ and $\mathbf{c}_{t_k}$ represent the normal and centroid of triangle t_k, likewise $\mathbf{c}_{b_1}$ represents the centroid of block b_1. Conversely, block b_0 is below the surface and has a negative sign ($\sigma_{b_0} = -1$) because the inner product $(\mathbf{c}_{t_j} - \mathbf{c}_{b_0}) \cdot \mathbf{n}_{t_j}$ is positive. This 'projection onto normal' approach provides the first method for block tagging.

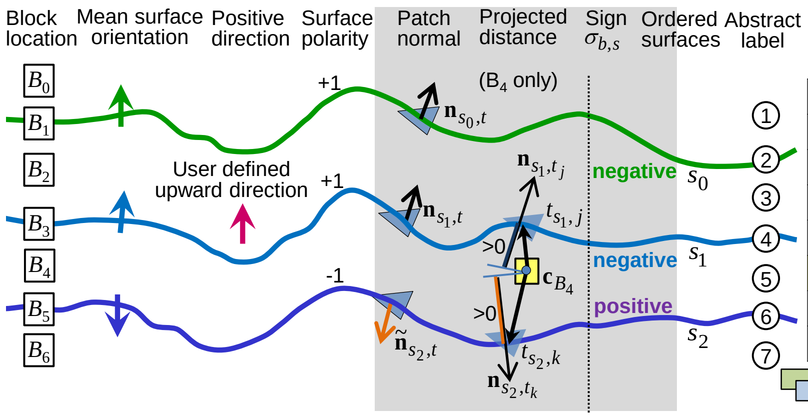
3.4.1 Discussion

For this method to work, the edges for each triangle in the mesh surface must be ordered consistently (e.g., in the anti-clockwise direction) and any ambiguity in regard to surface orientation must be resolved to ensure the assigned labels ultimately conform to user's expectation—e.g., the positive direction points upward with respect to an open surface (see below) or outward in the case of a closed surface.

3.4.2 Labelling scheme: extension to multiple surfaces

This tagging logic may be extended to multiple surfaces. Figure 7 depicts a multi-surface scenario and illustrates how abstract labels are assigned to multiple layers according to a logic table. Proceeding from left to right, blocks from seven distinct locations relative to the surfaces are shown. The *mean surface orientation* shows the direction obtained by averaging over all triangular patch normals. This arrow can point up or down provided the ordering of triangle vertices is consistent. Consistency can be enforced by ensuring triangle edges are traversed only in the anti-clockwise (or clockwise) direction.

The *positive direction* is a user-defined concept. By default, it points in the upward direction (+z axis). For a vertical surface (e.g., a geological feature such as a dyke) that runs across multiple layers, the positive direction may be defined as left (or right). *Surface polarity* indicates if there is agreement between the mean surface orientation (a property conferred by the mesh) and the positive direction (intention of the user). If they oppose, as is the case for the bottom surface in Figure 7, the polarity is set to negative. The significance is that the interpretation of the *projected distance* between a block and relevant triangle, and subsequently what sign we assign to the block, depend on the surface polarity.



Label assignment logic

block	σ_{b,s_0}	σ_{b,s_1}	σ_{b,s_2}	label
B_0	+1	+	+	1
B_1	0	+	+	2
B_2	–1	+	+	3
B_3	–	0	+	4
B_4	–	–1	+	5
B_5	–	–	0	6
B_6	–	–	–1	7

🟩🟦🟪 Affiliated surface s_n with respect to block location

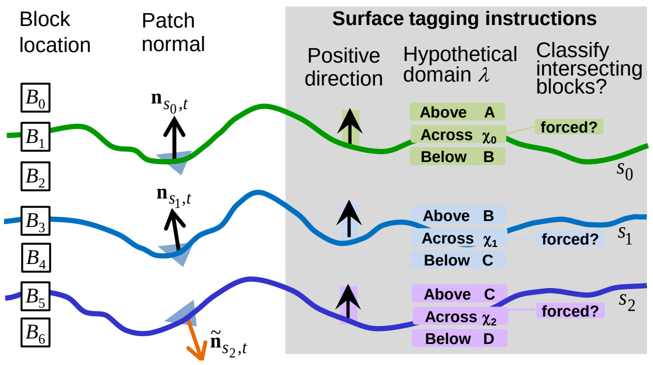
Figure 7: Block tagging in a multi-surface scenario: terminologies and label assignment logic.

Focusing now on the shaded portion of Figure 7, comparison of block B_4 with surface 1 yields $(\mathbf{c}_{t_{s_1,j}} - \mathbf{c}_{B_4}) \cdot \mathbf{n}_{s_1,t_j} > 0$, hence its sign σ_{B_4,s_1} is *negative* (it is below the s_1 surface). However, comparison with surface 2 yields a *positive* sign σ_{B_4,s_2} (it is above the s_2 surface) even though the projected distance $(\mathbf{c}_{t_{s_2,k}} - \mathbf{c}_{B_4}) \cdot \mathbf{n}_{s_2,t_k} > 0$; this is correct since s_2 has negative surface polarity which negates the logic. This method is adequate for simple surfaces such as those depicted in Figure 7; however it can fail in more complex situations. Therefore, the *projection-onto-patch-normal* approach is used here primarily as a vehicle for illustrating concepts and vulnerabilities. The pitfalls and robust solutions will be discussed in Section 5.

Abstract labels are assigned to merged blocks to distinguish between boundaries and embedded layers. In this example, layers are given odd-integer labels whereas boundaries (the interface between layers) are given even-integer labels, as indicated by the circles in Figure 7. For the interleaved layers, given its sign σ_{b,s_n} and affiliated surface s_n, the abstract label is given by $\lambda(s_n, \sigma_{b,s_n}) = 2 \times (n+1) - \sigma_{b,s_n}$ for $n \geq 0$, where $\sigma_{b,s_n} \in \{-1, 0, +1\}$ represent {below, across, above} the surface, respectively. The affiliated surface s_n is deduced from the shaded σ_{b,s_n} column of the logic table in Figure 7. For instance, at block locations B_3 and B_4, the affiliated surface is s_1. With $\sigma_{b,s_0} = -1$ and $\sigma_{b,s_2} = +1$ for $b \in \{B_3, B_4\}$, the label only depends on σ_{b,s_1} when $n = 1$.

3.4.3 Tagging instructions

With block tagging, there is tremendous scope for creativity. The scheme described below provides a flexible framework for domain specification given arbitrary surfaces. The diagram shown in Figure 8 is similar to Figure 7 for the most part. What is different is the emphasis on *surface tagging instructions*. Observe that there are three sets of tagging instructions: one for each surface. Each instruction specifies (1) a positive direction for each surface; (2) nominal labels for blocks located above, across and below each surface; (3) an override field (see "forced?") which communicates an intent to either **preserve** the surface intersecting blocks (by assigning a label different to any other layers) or **resolve** whether these blocks are strictly above or below the surface, i.e., force a binary decision.



Tagging instruction

Block	Value of λ			forced?
	negative	zero	positive	true
Interpretation				
	retain existing label	use abstract label	use specified value	classify as above or below
Label assigned to blocks				
B_0	not modified	1	A	A
B_1		2	χ_0	A/B
B_2		3	B	B
B_3		4	χ_1	B/C
B_4		5	C	C
B_5		6	χ_2	C/D
B_6		7	D	D

Figure 8: Block tagging instructions: interpretation of the lambda value.

A trivalent logic is built into the nominal labels specified in item 2. In Figure 8, this is denoted by λ. User-specified values are assigned to blocks when $\lambda > 0$. In addition, there are two special cases worth mentioning. First, an input value of $\lambda = 0$ instructs the program to use abstract labels instead of specific domain values (see "zero" column in Figure 8). Second, a negative input value $\lambda < 0$ instructs the program to **leave current labels intact**. This retains any prior label which has been assigned to that block (see "negative" column in Figure 8). This 'retain existing labels' mode allows the spatial structure of a block model to be updated without invalidating previous domain assignments. For instance, a surface

s_0 may be used as an upper bound. If $\lambda_{s_0}^{(\text{above})} = -1$, then all blocks located above this surface will not have their labels modified. Similarly, s_1 may be used as a lower bound. If $\lambda_{s_1}^{(\text{below})} = -1$, then all blocks located below this surface will not have their labels modified.

3.4.4 Retaining existing domain labels

As a motivating example, suppose we rotate the picture in Figure 8 counter-clockwise by 45 degrees. Further, suppose the entire block model is currently labelled as domain A. Then, the space between the two tilted surfaces (below s_0 and above s_1) can be used to model a dolerite channel that runs diagonally across the layers. By using the following specification: $\lambda_{s_0}^{(\text{above})} = -1$, $\lambda_{s_0}^{(\text{below})} = \text{B}$, $\lambda_{s_1}^{(\text{above})} = \text{B}$, $\lambda_{s_1}^{(\text{below})} = -1$ and forced(s) = TRUE for both, blocks within the embedded layer will be tagged as domain B (dolerite) for some B >0.

3.5 General interpretation

A volumetric block region (one comprising blocks of varying size) may be constructed in multiple passes and labelled sequentially by a single or a combination of surfaces using logical AND/OR operators based on the plane partitioning property of open surfaces, or expressed as the intersection/union of multiple closed surfaces. As an example, a fault surface may partition a space into two disjoint sets, say, blocks to the left (respectively, right) of the divide. Input blocks in contact with the fault line may be partitioned (subblocked) to conform to the shape of the fault surface. Subsequently, block tagging is applied independently to these two sets. The bounding surfaces that correspond across the discontinuity will provide similar block labelling instructions despite their physical separation. This ensures blocks within the same layer will receive the same label notwithstanding the vertical offset introduced by the fault.

4 Visualization

This section illustrates some of the results produced by the proposed framework. Figure 9(a) shows a regular block structure which covers the region $x \in [1000, 1250], y \in [750, 925], z \in [590, 670]$ with parent blocks measuring $25 \times 25 \times 5$m. Three horizontal surfaces are also given as input, these are shown in Figure 9(b). Surface-intersecting parent blocks identified in (c) are subject to block decomposition. Using block-triangle overlap detection, the surface-intersecting sub-blocks, each measuring $5 \times 5 \times 1$m, are identified in Figure 9(d). To reduce fragmentation, the decomposed sub-blocks in the surface-intersecting and non-intersecting sets are consolidated in Figure 9(e)–(f). For the surface-intersecting set, the block-count decreases from 8497 to 2102 after sub-blocks are coalesced.

In regard to the *forced* option for block tagging, the differences between preserving the boundary and classifying the blocks at the interface as strictly above / below a surface are demonstrated in Figure 10(g)–(h). For clarity, Figure 10(i) shows only blocks labelled as A and C (belonging to two domains of interest) in isolation. Clearly, they conform to the shape of the relevant surfaces. In Figure 10(j), only blocks that intersect with the top surface (labelled 2) have been extracted.

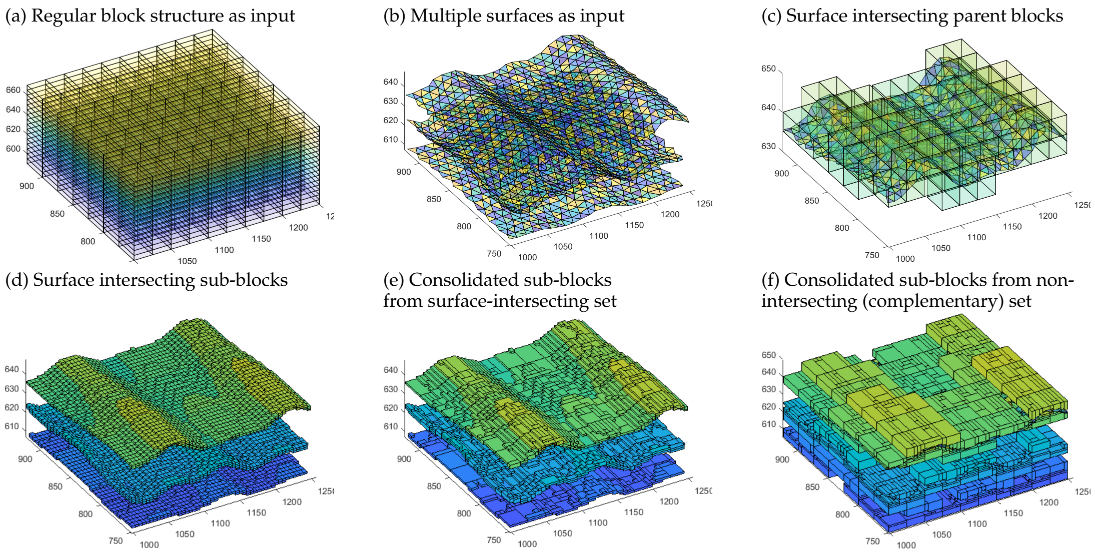

(a) Regular block structure as input
(b) Multiple surfaces as input
(c) Surface intersecting parent blocks
(d) Surface intersecting sub-blocks
(e) Consolidated sub-blocks from surface-intersecting set
(f) Consolidated sub-blocks from non-intersecting (complementary) set

Figure 9: Block restructuring given multiple surfaces.

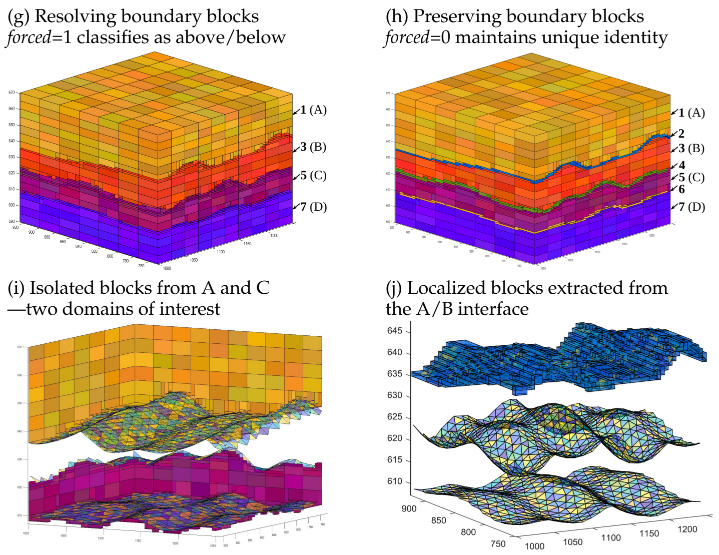

(g) Resolving boundary blocks *forced*=1 classifies as above/below
1 (A)
3 (B)
5 (C)
7 (D)
(h) Preserving boundary blocks *forced*=0 maintains unique identity
1 (A)
2
3 (B)
4
5 (C)
6
7 (D)
(i) Isolated blocks from A and C —two domains of interest
(j) Localized blocks extracted from the A/B interface

Figure 10: Block tagging given multiple surfaces.

4.1 Iterative refinement: an application to tilted surfaces

The running example has thus far taken only a regular block structure as input. In this section, we demonstrate that the framework can also modify the spatial structure of a model with irregular (non-uniform) block dimensions. Of particular interest is that tilted surfaces are used to model a hypothetical dyke channel running through bedded layers. This highlights two significant features: (a) ability to iteratively improve the spatial structure of an existing block model whilst preserving the labels for horizontal strata which have been previously assigned; (b) ability to work with oblique surfaces and produce correct result when the surface orientation is ambiguous (positive may not point upward), thus user has to specify precisely what is meant by the positive direction in relation to the supplied tagging instructions. An example of this is shown in Figure 11.

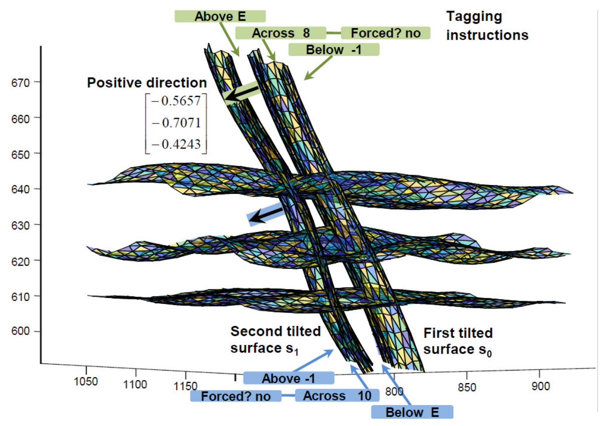


Figure 11: Block tagging instructions for tilted surfaces. The labels E, 8, and 10 refer to entities shown in Figure 12.

Figure 12 shows the recut block model follows the curvature of the tilted surfaces. In (k), blocks within the dyke are removed for clarity. Pre-existing labels (outside the dyke) remain intact. The spatial structure is only modified around the tilted surfaces. In Figure 12(l)–(m), blocks within the dyke and those located on the east / west interfaces are shown in isolation. Figure 12(n)–(p) show the existing labels for blocks in A, B, C, and D have been perfectly preserved. As expected, new labels—{8, 10}, and E respectively—have only been assigned to blocks that intersect with and sandwiched between the tilted surfaces.

5 Engineering perspectives and applications

Critical reflection and user feedback are both essential to designing robust and flexible systems [2]. Guided by the principle of reflective and iterative design [27], there was an early and continued focus in our approach on real usage scenarios [25], as well as successive evaluation, modification, and scenario-based testing. In this section, we describe two improvements made to eliminate flaws identified through this process which made the system more robust.

5.1 Issue 1: Sign inversion due to sparse, jittery surface

In regard to block tagging and 'which-side-of-the-surface' determination, the 'projection-onto-normal' method described in Section 2.4 works well when the surface is smooth and triangle mesh is dense and uniform. Potential issues arise when the surface exhibits local

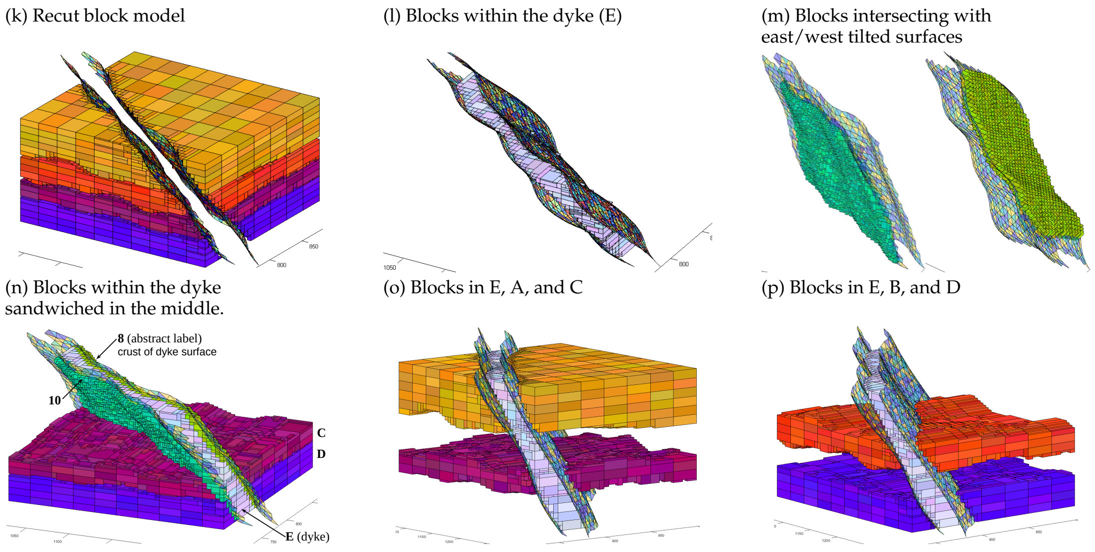

(k) Recut block model
(l) Blocks within the dyke (E)
(m) Blocks intersecting with east/west tilted surfaces
(n) Blocks within the dyke sandwiched in the middle.
8 (abstract label) crust of dyke surface
10
C
D
E (dyke)
(o) Blocks in E, A, and C
(p) Blocks in E, B, and D

Figure 12: Iterative refinement incorporating tilted surfaces.

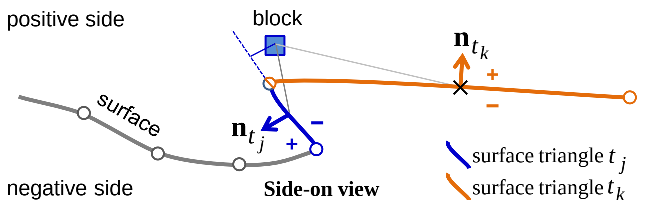


Figure 13: Local sign inversion when triangle mesh surface is sparse and jittery.

jitters and the triangles are sparse. For instance, the mesh resolution is low relative to the parent block size when triangle patches stretch over distances of up to one kilometer in certain areas. As an illustration, Figure 13 shows a block associated with triangle t_j (the nearest patch based on block-triangle centroid-to-centroid distance). This association yields the wrong result, a negative sign with respect to the normal $\mathbf{n}_{t_j}$ is obtained (according to the plane partitioning test) even though it lies above the surface. We observe the projection of the block lies outside the support interval of the referenced triangle t_j and the same comparison with t_k which is further away would produce the right result (a positive sign). This problem can be remedied by upsampling the mesh surface to increase its density. Figure 14 shows an example where triangles are recursively split along the longest edge until the maximum patch area and length of all edges fall below the thresholds of 1250m^2 and 100m. A better solution, however, is ray-tracing [21]. In general terms, ray-tracing determines whether a block is above or below an open surface (resp., inside or outside a closed surface) by counting the number of intersections between the surface and a ray casted from the block. This recommended approach is described in Appendix B. The key advantage is that ray tracing is not susceptible to variation in surface mesh density (the sign inversion problem due to folding), furthermore, it does not require dense surfaces or consistent (e.g., clockwise) ordering of triangle vertices. Therefore, ray-tracing will be the method of choice for block tagging moving forward. It replaces the projection-onto-normal approach in our proposal.

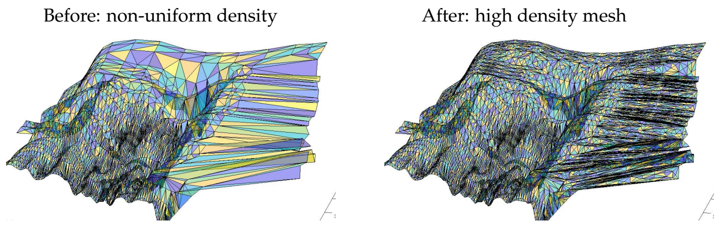


Figure 14: Increasing mesh density of a real surface by splitting recursively the longest edge of triangles whose area is above the threshold.

5.2 Issue 2: Boundary localization accuracy

The block consolidation component, as it currently stands, considers the sub-blocks that belong to the surface-intersecting set ($\mathcal{B}_{\text{intersect}}$) and non surface-intersecting set ($\mathcal{B}_{\text{non-intersect}}$) independently. As Figure 5 has shown, the sub-blocks (cells) within each respective set are merged separately to form larger rectangular prisms within the confines of the parent blocks. While this split is useful for extracting surface-intersecting sub-blocks, it has two drawbacks. In terms of **boundary localization accuracy**, Figure 15(a) shows that surface-intersecting sub-blocks—with centroids located on different sides of the surface—are merged together irrespective of whether it is predominantly above or below the surface. In terms of compaction, **merging potential is limited** because adjacent cells from $\mathcal{B}_{\text{intersect}}$ and $\mathcal{B}_{\text{non-intersect}}$ cannot be coalesced even if they lie on the same side of the surface.

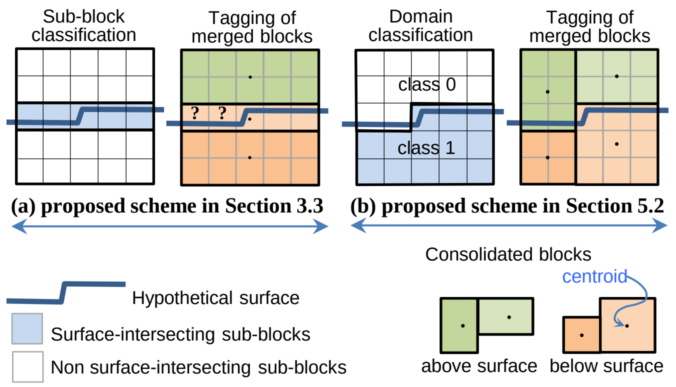


Figure 15: Treating surface-intersecting and non-intersecting sub-blocks, $\mathcal{B}_{\text{intersect}}$ and $\mathcal{B}_{\text{non-intersect}}$, independently during block consolidation may reduce boundary localization accuracy and limit merging potential. In the latest proposed scheme (Section 5.2), sub-blocks are classified by their location relative to the surface before sub-blocks consolidation.

To reinforce the first point, the two cells marked with "?" in Figure 15(a) ought to be labelled as above the surface. However, under the current regime, they are considered jointly with the three cells immediately to the right that also intersect with the surface, thus they are treated collectively as a 1-by-5 merged block. Since the centroid (black dot) lies marginally below the surface, the merged block will also be labelled as such. This distorts the boundary as it introduces a vertical bias of around $\Delta^{(\text{block})}_{\text{y,min}}/2$ to the two left-most cells. To accurately localize the boundary and achieve the result shown in Figure 15(b), ray tracing is used to determine the location of cells with respect to each relevant surface that intersects the parent block. Given S surfaces, the {0=above (or no intersection), 1=below} decisions naturally produce up to 2^S categories (or states) which would be treated separately during sub-blocks consolidation in lieu of $\mathcal{B}_{\text{intersect}}$ and $\mathcal{B}_{\text{non-intersect}}$. In practice, however, we suggest encoding the location with respect to each surface s using 3 bits $b_{3s+2}b_{3s+1}b_{3s}$, where the mutually exclusive bits are set to 1 to denote the outcomes of *above*, *below*, and *untested*, respectively. The rationale is that when $b_{3s} = 0$, domain identification needs not be attempted during block tagging with respect to surface s, since the decision has already been made here prior to sub-blocks consolidation (either $b_{3s+2} = 1$ or $b_{3s+1} = 1$) and this information is passed on. In fact, applying ray-casting at the cellular-level (highest resolution) yields more accurate results near the surface than applying to merged blocks, especially for undulating surfaces with high local curvature.

To summarize, administering ray-casting before sub-blocks consolidation helps divide cells along surface boundaries; this increases boundary localization accuracy and ensures merging is performed within the right domains with maximum potential. Table 1 provides a comparison of the techniques discussed.

Techniques for 'which side of the surface' determination
• **Projection onto normal method** (Section 2.4)
+ Ability to extrapolate boundary beyond surface support interval
– Susceptible to sign inversion problem depicted in Fig. 13
• **Ray tracing method** (Section 5.1 and Appendix B)
+ Not impacted by sparse, jittery surface
+ Consistent edge ordering in triangular mesh surface not required
– Result only defined over support interval of the supplied surface

Techniques for sub-blocks consolidation and boundary localisation
• **Earlier proposal** (Section 2.3)
– Limited block merging potential due to class segregation
– Possible boundary distortion (introduce small bias)
• **Latest proposal** (Section 5.2)
+ Classify sub-blocks w.r.t. surface before consolidation
+ Accurate (merge & label decisions made at highest resolution)

Table 1: Comparison of techniques with emphasis on system robustness.

5.3 Applications

The first application we preview relates to block modeling for a typical mine site located in Western Australia. The block model spatial restructuring results shown in Figure 16 illustrate a block-wise partitioning of the mine site into different domains. This is achieved using surfaces which were created to separate geological domains (mineralized, hydrated, and waste) within the Brockman Iron Formation which contains members of interbedded

BIF and shale bands in the Hamersley Basin Iron Province [5]. Although ore-genesis theories vary depending on the minerals or commodity, the ability to model formations and features such as igneous intrusions in ore deposits is of general interest in areas not limited to mining, but also in further understanding the structural geology of mineral deposits. Using open and closed surfaces to represent structures of varying complexity—this may encompass volumes with exceptional geochemical or geophysical attributes—it is possible to extract waste pockets with high concentration of trace elements, or regions with magnetic / gravity anomaly [32]. Equally, if the surfaces represent the boundary of aquifers separated by clay and lignite seams [34], the process may serve as a basis for creating a structural hydrogeological model to study hydraulic and transport conditions in geotechnical or environmental risk assessment. The techniques developed for shaping a 3D block model can be used potentially in a variety of contexts, including surface buffer analysis (in GIS and structural modeling) for triangle mesh 3D boundary representation of localized objects [16]. In the next section, we focus on a specific application of block merging to reduce spatial fragmentation in a block model.

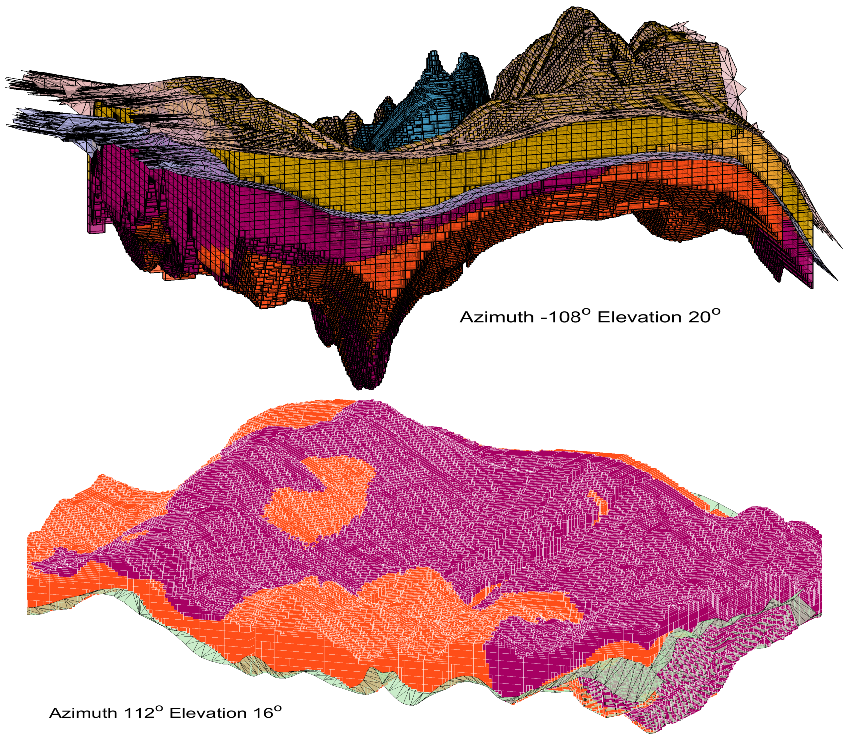


Figure 16: Typical block model spatial restructuring results. Top: blocks partitioned by surfaces into different domains (not all surfaces are shown). Bottom: reveals two block sets with different levels of mineralization in an ore deposit. In the PDF article, zoom in to see individual blocks.

6 Block merging to reduce spatial fragmentation

The proposed block merging algorithm can also be used to consolidate a fragmented block model that exists with or without reference to any mesh surface. Spatial fragmentation is used in this context to mean a highly redundant block model representation where blocks near the boundary are over-segmented or divided in an excessive manner to follow the curvature of a surface without regard for the compactness (total block count) of the model. As an illustration, Figure 17(b) shows a highly fragmented block model for the *Stanford Bunny* created by block decomposition without consolidation. In an effort to closely approximate the surface, numerous blocks at the minimum block size were produced near the surface. Figure 17(c) shows a clear reduction in block density as blocks are appropriately merged. This results in a more compact block representation (3D segmentation) of the object.

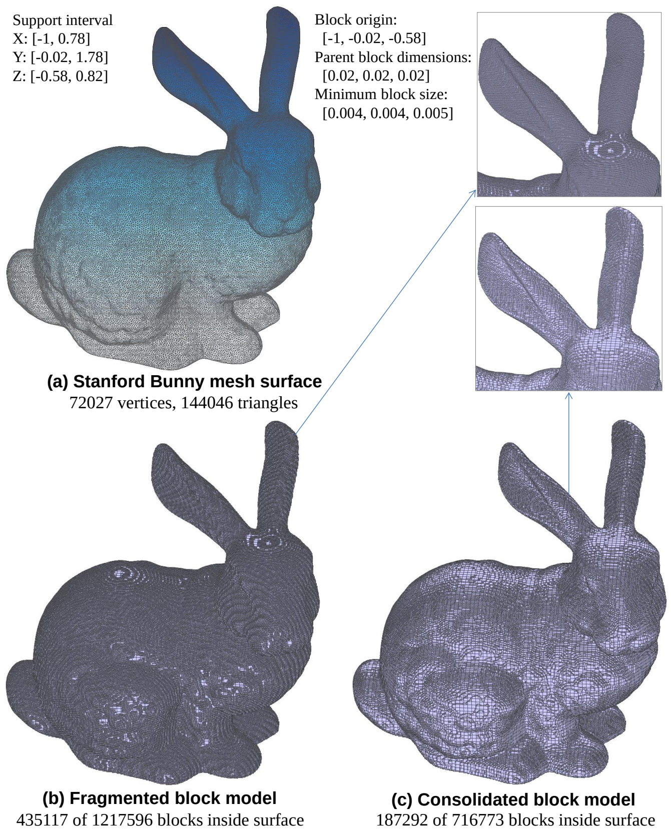

Support interval
X: [-1, 0.78]
Y: [-0.02, 1.78]
Z: [-0.58, 0.82]
Block origin:
[-1, -0.02, -0.58]
Parent block dimensions:
[0.02, 0.02, 0.02]
Minimum block size:
[0.004, 0.004, 0.005]
(a) Stanford Bunny mesh surface
72027 vertices, 144046 triangles
(b) Fragmented block model
435117 of 1217596 blocks inside surface
(c) Consolidated block model
187292 of 716773 blocks inside surface

Figure 17: Block merging applied to Stanford Bunny to reduce block fragmentation. Zoom in to see individual blocks.

The block merging algorithms are formally described in the Supplementary Material wherein a number of technical issues are discussed in depth. At its core, a notable feature is that 'feasibility of cell expansion' is determined using a multi-valued (rather than boolean) 3D occupancy map, where the values correspond to the identity of the cells or sub-blocks. The spatial constraints governing cell expansion are somewhat different, in particular, the lateral dimensions orthogonal to the axis of expansion have to match for all the blocks involved in a merge. These, along with other relevant considerations and implementation details, are described in Appendices D.4–D.6.

6.1 Merging conventions and optimization objectives

The block merging algorithms also recognize that merging can be performed under different conventions. For example, in Algorithm 2, the procedure preserves the input block boundaries, it does not introduce new partitions (sub-divisions) that are not already present in a parent block. This merging convention is referred as **persistent block memory**. It has the property that each input block is mapped uniquely to a single block in the merged model. In contrast, Algorithm 1 implicitly erases the sub-block boundaries before block consolidation begins. This merging convention is referred as **dissolved sub-block boundaries**, it generally achieves higher compaction because it makes no distinction between input blocks from the same class and parent. It is able to grow blocks more freely and produce fewer merged blocks since the size compatibility constraints between individual blocks no longer apply when internal sub-block boundaries are ignored. These differences are illustrated in Figure 18. The *dissolved sub-block boundaries* convention can be useful for

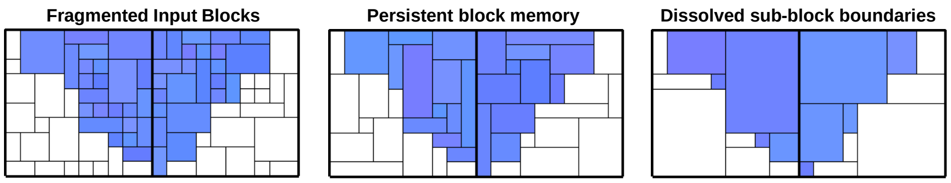


Figure 18: Differences between two block merging conventions: persistent block memory versus dissolved sub-block boundaries in 2D.

healing a fractured block model, for instance, over a region where a false geological boundary was given in a previous surface update. Under the *dissolved sub-block boundaries* convention, coordinate-ascent can start from a clean slate and merge sub-blocks in a fragmented area back to the fullest extent in cases where individual sub-block dimensions or internal boundary alignments are otherwise incompatible. It is not bound by the consequences of prior model restructuring decisions. The technical details are given in Appendix D.7.

For readers simply looking for a basic explanation, Figure 19 reinforces these points and illustrates what *healing* really means in practice. Figure 19(a) shows an existing (poorly estimated) boundary created using limited data at an earlier point in time. Figure 19(b) shows the existing block model structure created using this surface which unbeknown to the user is not faithful to the actual boundary shown in gray. At a later point in time (perhaps after several months have elapsed), more assay measurements have been taken at previously unsampled locations. These denser observations help improve the estimated boundary contact points and a new (more accurate) surface is produced as a result; see bottom curve in (a). A second iteration of the block model spatial restructuring process is applied using the new surface. This workflow significantly improves the boundary localization property of the new block model in (c), however it leaves behind remnant sub-blocks from the previous iteration. Relaxing the internal constraints, block merging under the *dissolved sub-block boundaries* convention helps promote healing in fractured areas. As evident in (d), fragmented blocks are coalesced into larger blocks in areas where the misplaced boundary (blue surface) once occupied.

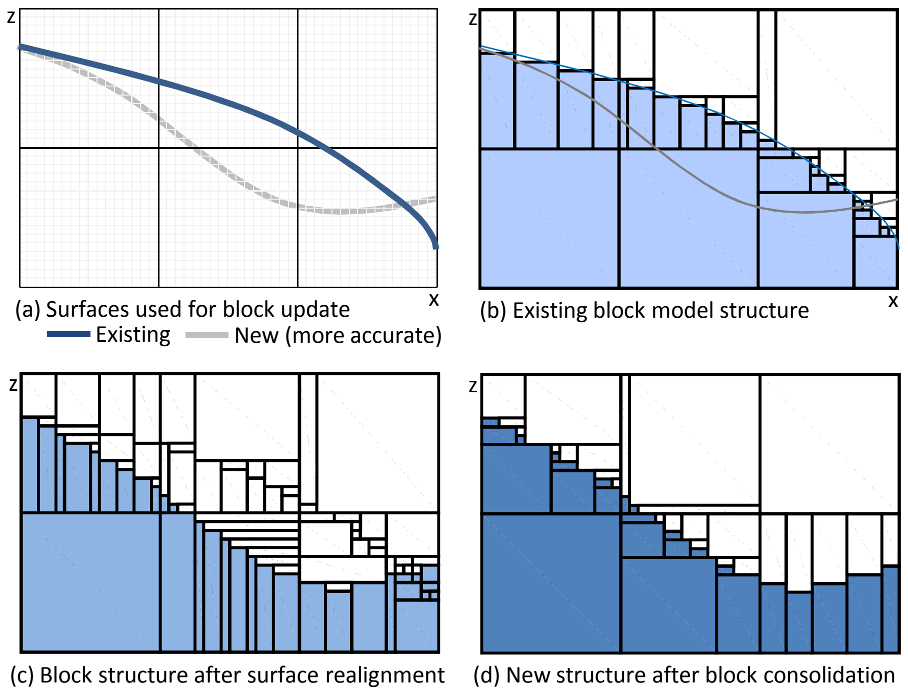

z
x
(a) Surfaces used for block update
Existing
New (more accurate)
(b) Existing block model structure
(c) Block structure after surface realignment
(d) New structure after block consolidation

Figure 19: Example of boundary healing through block consolidation—refer to description in main text.

The final block merging strategy (see Algorithm 3 in Appendix E) can optimize results with respect to different objectives. For instance, it can minimize the block count or avoid extremely elongated blocks by optimising for the block aspect ratio. It also exploits symmetry by following different scan patterns to alleviate ordering effects associated with using a fixed starting point. These issues are further discussed in Appendices D.9 and D.10.

6.2 Pseudocode

The block merging algorithms are formally described in Appendix E which constitutes part of the Supplementary Material.

6.3 Evaluation

Two experiments were performed targeting two different usage scenarios. The first experiment described in Section 7 uses only the block merging algorithms to reduce block fragmentation in an existing block model where mesh surfaces are not available and the minimum block size has been fixed by the supplier of the model a priori. This restricted setting requires only the use of the sub-blocks consolidation component depicted in the system diagram (see Figure 2). The main objective is to analyse its performance in terms of block compaction and multi-threaded execution.

The second experiment described in Section 8 provides an end-to-end evaluation of the entire system (from block model creation to block tagging) which applies sub-block consolidation and ray-tracing in an iterative setting. This involves over 80 surfaces and creates approximately 30 different domains. The goal is to compare the proposed strategies with two hierarchical subblocking techniques based on octrees, to highlight the constraints of dyadic decomposition and the importance of inter-scale block merging.

7 Experiment 1: Block merging efficacy on a mining resource block model

The proposed block merge strategy (Algorithm 3) was implemented in C++ (with boost python bindings) and evaluated using a real model developed for a Pilbara iron ore mine in Western Australia. The model contains 697,097 input blocks of varying sizes[3] spanning 1342 parent blocks with dimensions $(200, 200, 20)$. The input block model is given as is without associated surfaces. The minimum block dimensions are also fixed by the user at $(5, 5, 1)$. A key requirement is that blocks smaller than $(5, 5, 1)$, or being some fractional multiples of it, must not be introduced in the output. This is guaranteed by the proposed system.

The metrics of interest are the block aspect ratio, merged block count and execution times with multi-threading. The *AspectRatio*(m) obtained using method m is defined by $f_{\pi^*}(\{\mathbf{\Delta}^{(b,\pi^*)}\}_{b\in\mathcal{S}_{\mathrm{P},\lambda}})$ in (D.2). For comparison, a python implementation that uses a greedy merging strategy based on block edges connectivity is chosen to establish a baseline. Its sub-blocking approach is based on prime number factorization. Given parent blocks with cell dimensions $(k_x, k_y, k_z) = (40, 40, 20) = 2^2 \cdot 5 \times (2, 2, 1)$, it seeks to grow cells into subblocks with dimensions 2^n for $n \in \mathbb{Z}$ or 5 and merge iteratively.[4]

7.1 Statistical perspective

Figure 20 illustrates the block merging performance of Algorithm 2 versus the baseline with emphasis on block aspect ratios. In this plot, performance is measured in terms of block aspect ratio. The volume-weighted aspect ratio is computed for each parent block and sorted in ascending order. Large aspect ratios correspond to long narrow blocks whereas small aspect ratios indicate more balanced, less extreme block dimensions which are more preferred in a mining context. In this plot, the lower the curve, the more desirable it is. The main observation is that the proposed block merge strategy is superior to the baseline. Even when the coordinate-ascent procedure follows only the standard scan pattern, it produces higher quality merged blocks (dotted line is below the dash line). The performance margin increases significantly when multiple scans are deployed in Algorithm 3 to minimize the aspect ratio explicitly (note: solid line is consistently below the dash line).

[3]There were 200 unique input block dimensions, these range from $(5, 5, 1)$ to $(50, 50, 2)$ with x and y varying in increments of 5.

[4]This heuristic is clearly suboptimal since 3 does not appear in the prime factorization of the cell dimensions, it is not possible for sub-blocks with k_x, k_y, k_z of 3 to emerge in the first pass.

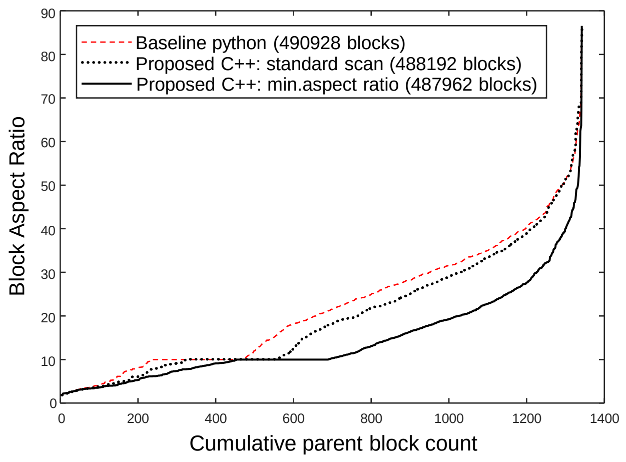


Figure 20: Inverse cumulative distribution of volume-weighted block aspect ratio versus parent block count.

7.2 Spatial perspective

Figure 21 provides an alternative view of the same result from a spatial perspective. A birds eye view and two cross-sections from the *proposed* method are shown. Blocks are colored to highlight differences in aspect ratio between the baseline and proposed method. The darker the red, the more superior is the proposed method. Conversely, blue blocks show the proposed method is inferior. The dominance of the red blocks show the proposed block merge method is able to consistently produce blocks with less extreme aspect ratios across the site. A magnified view for two regions of interest are shown in Figure 22.

7.3 Local perspective

To verify these results, we compare the baseline and proposed method for parent blocks with a high contrast in $\log\left(\frac{AspectRatio(\text{baseline})}{AspectRatio(\text{proposed})}\right)$. Figure 23 shows the sub-block structure within a parent block of interest, it shows from left to right: the input blocks, and merge results from the baseline and proposed methods. The key observation is the presence of thin narrow blocks in the baseline result; these disappeared under the proposed method. Algorithm 3 discovers block merging opportunities missed by the baseline method. It is able to minimize the block aspect ratio by conducting multiple scans during coordinate-ascent.

Figure 24 shows a second example. In the top panel, blocks are colored by block volume such that bigger blocks are colored in warm colors (yellow, for instance). This shows the proposed method is more effective at consolidating smaller blocks into larger blocks. The bottom panel shows the merge result for the same parent block from a different vantage point. Here, blocks are colored by domain labels. This illustrates an interesting case where there is a limit to how much merging can be achieved because input blocks from different classes are processed independently even when they share the same parent.

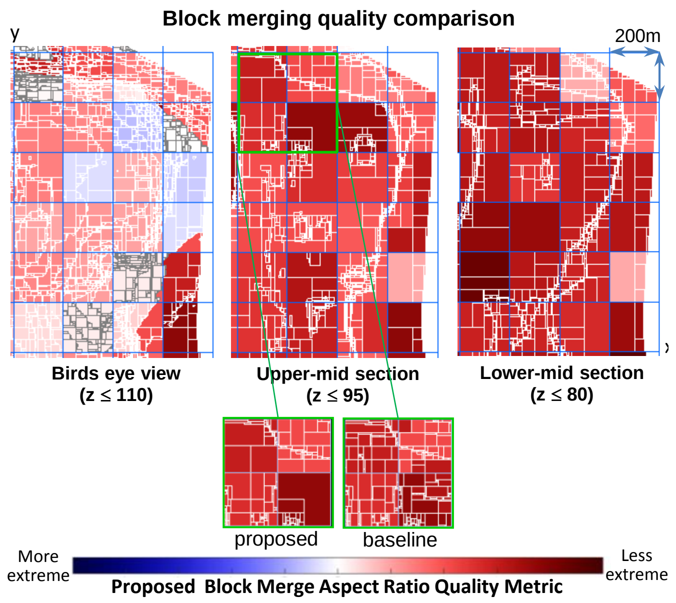


Figure 21: Cross-sections of merged block model colored by contrast in block aspect ratio.

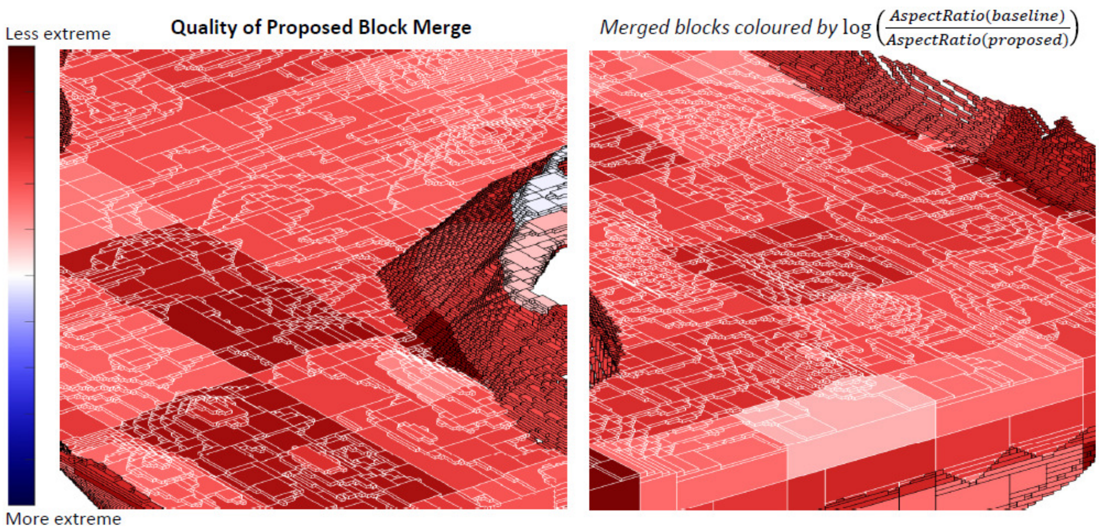

Less extreme
Quality of Proposed Block Merge
More extreme
Merged blocks coloured by log (AspectRatio(baseline) / AspectRatio(proposed))

Figure 22: Spatial perspective of merged block model in regions of interest.

7.4 Block compaction

Table 2 reports the number of output blocks produced by the baseline and proposed methods. The *block count* column shows the proposed method is more efficient at coalescing blocks than the baseline method, as the block count under "persistent block memory" is lower. The block count reported for the proposed method under "dissolved sub-block

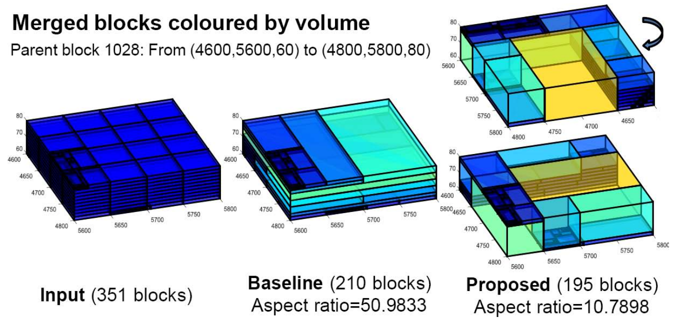


Figure 23: Block merging local comparison 1—an instance where the log-contrast in block aspect ratio, $\log(AR_{\text{baseline}}/AR_{\text{proposed}})$, is highest amongst all parent blocks.

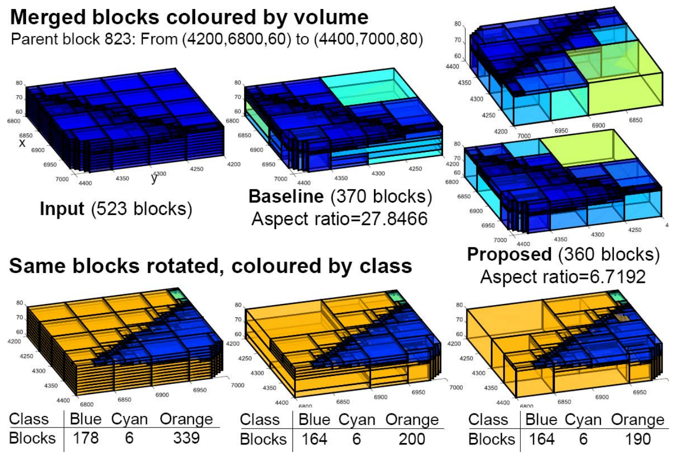


Figure 24: Block merging local comparison 2—a second instance where the log-contrast in block aspect ratio is amongst the highest over all parent blocks.

boundaries" is lower still, this is consistent with our expectation and the reasoning given in Appendix D.7 based on constraint relaxation.

merge method	convention	block count	relative %
Input	–	697,097	100.0%
Baseline	–	490,928	70.42%
Proposed	persistent †⋆	487,962	70.00%
Proposed	dissolved ‡⋆	447,412	64.18%

† = persistent block memory ‡ = dissolved sub-block boundaries
⋆ both using multiple-scans, minimising block aspect ratio

Table 2: Block merging given irregular blocks: output block count comparison.

7.5 Execution times and memory

Table 3 reports the processing time for block merging and demonstrates the scalable nature of the proposed algorithm.[5] The parallel nature of the algorithm is a consequence of processing all input blocks that belong to the same parent block together independent of other parent blocks. This allows processing to be compartmentalized. From the description in Section 3.3, it is clear that block consolidation and block tagging are applied locally to spatially disjoint regions; this is exploited in a multi-threaded implementation.

Merging under the "dissolved sub-block boundaries" convention given 697,097 blocks (of varying sizes) from the input model, is equivalent to merging 29,154,116 blocks (with single cell dimensions) directly under the "persistent block memory" convention. The problem size of "*dissolved*"‡ grows by a factor of 41.8, however, the computation time increased sub-linearly by a factor of only 9.5, this is partly due to the removal of sub-block alignment / compatibility constraints when internal sub-block boundaries are dissolved. Of course, the run time still increases relative to "*persistent*", ultimately this may be viewed as a trade-off between speed and compaction when Table 3 is viewed along side Table 2. The CPU utilization and memory use columns are included for completeness, but these figures are otherwise unremarkable. Running 16 threads, the physical memory footprint amounts to 1% of the RAM available.

merge method	convention	threads	time (s)	relative	cpu%	mem (kB)
Baseline	–	–	254.181	1.0000	94.1	365,344
Proposed	persistent †$*$	1	4.870	0.0192	–	–
Proposed	persistent †$\star$	1	32.852	0.1292	100.0	285,076
Proposed	persistent †$\star$	4	9.840	0.0387	400.0	380,884
Proposed	persistent †$\star$	16	4.162	0.0164	1553.0	614,540
Proposed	dissolved ‡$\star$	1	469.462	1.8470	100.0	293,988
Proposed	dissolved ‡$\star$	4	125.111	0.4922	400.0	370,108
Proposed	dissolved ‡$\star$	16	39.675	0.1561	1547.0	609,064

† = persistent block memory ‡ = dissolved sub-block boundaries
$*$ = single scan, coordinate-ascent follows the standard pattern
$\star$ multiple scans, minimising block aspect ratio

Table 3: Block merging processing times and memory footprints comparison.

8 Experiment 2: The proposed coordinate-ascent block merging technique versus octree subblocking

Octree decomposition is a volumetric partitioning strategy that is well studied in the literature [26]. As this hierarchical structure has previously been considered as a potential representation for spatial information systems in geology [11], it provides an interesting reference point to measure our proposed method against. To obtain a fair comparison, the parent blocks ($\boldsymbol{\Delta}^{\text{block}}_{\text{parent}} \in \mathbb{R}^3$) must be divisible by the minimum block size ($\boldsymbol{\Delta}^{\text{block}}_{\text{min}}$) by some vector $\mathbf{k} = (k_{\text{x}}, k_{\text{y}}, k_{\text{z}})^T = (2^{K_{\text{x}}}, 2^{K_{\text{y}}}, 2^{K_{\text{z}}})^T \in \mathbb{Z}^3_+$ where each element is a power of two.

[5]Experiments were conducted on a linux machine with 32 CPUs, 8 cores (Intel Xeon® CPU E5-2680 @ 2.70GHz), 64GB RAM and 2G swap memory.

This dyadic constraint is necessary for compatibility since the standard octree has to split each dimension into half along the x, y, and z axes at each spatial decomposition step. The analysis presented earlier in Table 2 pertains to a block model where its parameters (e.g., $\mathbf{\Delta}_{\text{min}}^{\text{block}}$) are already fixed and the cell dimensions $\mathbf{k} = (40, 40, 20)^T$ are not divisible by 8; thus the maximum number of dyadic decomposition is limited to $D = 2$. To facilitate a more wide-ranging comparison, we have chosen another site for which the block structure is not fixed a priori and that geological surfaces are available. This enables all system components in Figure 2 including block model creation (not just block merging) to be applied. The chosen parameters $\mathbf{\Delta}_{\text{parent}}^{\text{block}} = (50, 50, 20)^T$ and $\mathbf{\Delta}_{\text{min}}^{\text{block}} = (1.5625, 1.5625, 0.625)^T$ satisfy the dyadic constraints and allow a maximum of $D = 5$ levels of octree decomposition. Figure 25 illustrates the complexity of the domain structure. This surface-induced structure provides spatial separation between significant geological units. A detailed explanation in terms of stratigraphy, material type or geochemical composition however is beyond the scope of this work. The main point is that deposits in the Hamersley Group typically contain interlayered Banded Iron Formations (BIF) and shale bands. Using multiple surfaces to express boundary constraints, it is possible to isolate, for instance, areas of localized iron enrichment within the BIFs where high grade iron ore deposits occur. Interested readers may refer to [29] for geological context.

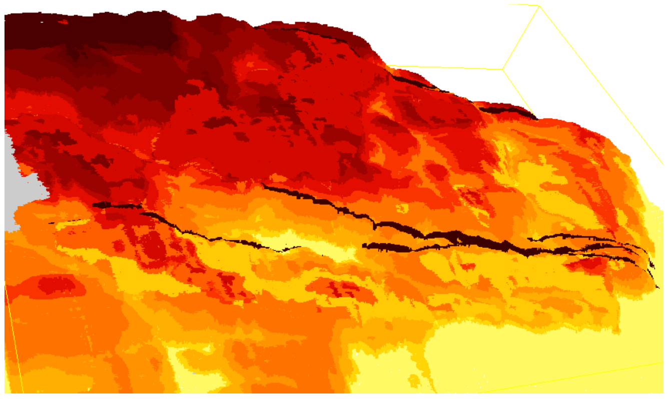


Figure 25: Spatial structure of a mine where block model comparison between the proposed and octree representation are made. Around 30 geological domains are represented at this site, each colored in a different shade. Black stripes represent igneous intrusions demarcated by a set of closed surfaces (not shown). The layers are peeled back one-by-one in an animated sequence in Appendix G. Please refer to pages 13–15 of the supplementary material.

The schemes being compared are the two proposed coordinate-ascent block merging algorithms performed under the *persistent block memory* and *dissolved sub-block boundaries* conventions (denoted 'Proposed-P' and 'Proposed-D' respectively) versus octree decomposition and a variant that permits intra-scale merging (denoted 'Octree' and 'Octree+Merge' respectively). In a standard octree decomposition, a heterogenous node (viz. a block with dimensions $\mathbf{\Delta}_d = \mathbf{\Delta}_{\text{parent}}^{\text{block}}/2^d$) that contains smaller blocks with different labels is split into an octant with $2 \times 2 \times 2$ cells of size $\mathbf{\Delta}_d/2$. This sub-division, similar to the approach implemented in the *Surpac* software [24], is applied recursively to all heterogenous nodes at each decomposition level $d = 1, 2, \ldots, D$. This process shatters a volume into many

	Total block count, B_D			
D levels	Octree	Octree + Merge	Proposed-P	Proposed-D
3	4241289	1830826	1094917	1100585
4	17414755	7455284	4053310	4007018
5	74065301	31052736	15608162	15204844

	Relative block count			
D levels	Octree	Octree + Merge	Proposed-P	Proposed-D
3	100%	43.167%	25.816%	25.949%
4	100%	42.810%	23.275%	23.009%
5	100%	41.926%	21.074%	20.529%

The total block volume is approximately 3.693382×10^9 m^3.
A detailed breakdown of these figures is given in Appendix G.

Table 4: Block model aggregate block counts: octree versus proposed methodology.

	Block count growth factor based on Table 4			
Growth rate	Octree	Octree + Merge	Proposed-P	Proposed-D
B_4/B_3	4.1060	4.0721	3.7019	3.6408
B_5/B_4	4.2530	4.1652	3.8507	3.7946
B_5/B_3	17.4629	16.9611	14.2551	13.8152

Table 5: Block model block count growth factors: octree versus proposed methodology.

smaller blocks and thus vastly increases the block count in its pursuit of higher accuracy. The 'Octree+Merge' variant allows 2 and 4 edge-connected cells to be consolidated into a single (rectangular or square-like) block thus achieves higher compaction. This octree merging step applies only to cells at the same scale within an octant in the spatial hierarchy. For completeness, a full description and pictorial examples of these concepts are given in Appendix F.

8.1 Block count analysis

In Table 4, it can be seen that performing octree decomposition without merging produces a large number of blocks (4241289 for $D = 3$ levels). With octree merging, the block count is reduced to 43.167%. The proposed strategies have a block count of about 25% relative to standard octree decomposition. It is worth mentioning that despite this block count disparity, each block model represents the modeled region with the same fidelity and the total block volume for each domain classification remains the same irrespective of D and the block count. In one sense, the 'Octree+Merge' method provides an empirical bound on block model efficiency (as measured by block count) that other strategies can surpass. The impressive efficiency gains achieved by the proposed strategies may be attributed to inter-scale block merging without dyadic constraints. For instance, it is possible to combine two adjacent blocks $\mathcal{B}_i$ and $\mathcal{B}_j$ in neighboring octants if their labels are the same and their cell-dimensions, say, $\mathbf{k}^{(i)} = (1, 2, 1)$ and $\mathbf{k}^{(j)} = (2, 2, 1)$ are compatible. This would not be possible using the naïve 'Octree+Merge' method as it involves blocks from different octants.

	weighted by block volume			
D levels	Octree	Octree + Merge	Proposed-P	Proposed-D
3	2.5	2.958828	3.668188	2.615373
4	2.5	2.968178	4.901418	2.941797
5	2.5	2.972378	7.395642	3.456381

Table 6: Block model average block aspect ratios: octree versus proposed methodology.

From Table 4, it is evident the block count roughly grows by a factor of 4 with each additional decomposition level. This is made clear in Table 5 and expressed as a ratio $B_d/B_{d'}$ between levels d and d'. Intuitively, the growth rate relates to the surface contact area with the blocks which scales with $O(n^2)$ where n represents the number of cells measured in one dimension. Interestingly, the proposed coordinate-ascent block merging strategies have a lower growth rate compared to 'Octree+Merge'—moving from 3 to 4 levels, the block count increases by a factor of 3.6408 as opposed to 4.0721; from 3 to 5 levels, the growth rate is 13.8152 as opposed to 16.9611. In this respect, the proposed strategies scale better and they are more efficient at representing the same domain information. We caution however that block modeling is not ultimately about an endless pursuit for increasing precision. Block granularity (which may be specified in terms of the number of decompositions, D) is often limited by practical considerations. In our application, individual blocks in the model serve as containers for grade estimation. The blockwise spatial distribution of chemical attributes in turn informs what, where and how ore material are excavated. The heavy equipments used to extract and transport such material operate with physical or kinematic constraints (such as bucket size and turn radius), so the required spatial resolution in the case of mining is probably limited to 1.5m to 6m at best. This naturally imposes a lower bound on the spatial fidelity needed from a model. It informs the choice of D in our evaluation, as $D=3$ and $D=5$ yield a resolution of 1.5625m and 6.25m, respectively, in a horizontal plane.

8.2 Block aspect ratio analysis

For grade estimation at a given level of spatial fidelity, a lower block count is desirable not so much for storage but from a computational perspective. A compact model contains fewer blocks that require the inference engine to estimate into, and by extension, it has fewer values to update when new information becomes available. For instance, the difference between 'Octree+Merge' and 'Proposed' can be as much as 50% (based on $D = 5$ in Table 4). A secondary consideration is an optimization preference for more uniform blocks (as opposed to thin, narrow blocks) that have lower aspect ratios. In a pure octree decomposition scheme, the blocks always maintain the same aspect ratio (in our experiment, this value is 50/20 or 2.5). However, this is obtained at significant cost, resulting in a substantial increase in block count. In Table 6, the main observation is that the weighted aspect ratios in the 'Proposed-D' column are a marked improvement over 'Proposed-P' when merging constraints are relaxed (when input block boundaries are no longer persistent in the sense defined in Section 6.1). It also increases slower than 'Proposed-P' as the resolution (number of levels D) increases, and tends toward the values in the 'Octree+Merge' column.

8.3 Block fragmentation analysis

The diversity of blocks within each model is examined in Figure 26. The graph shows the cumulative distribution of block dimensions for $D = 3$ decomposition levels, with block volumes arranged in ascending order (from left to right) and sorted by aspect ratio in the event of a tie. The staircase nature of the orange curve for 'Octree+Merge' shows that there is a much smaller pool of possible block size permutations produced via octree decomposition with merging. To simplify the analysis, we measure blocks in multiples of $\mathbf{\Delta}_{\text{min}}^{\text{D=3}} = (6.25, 6.25, 2.5)$, for instance, using $\mathbf{k} = (2, 1, 3)$ to denote a block with real dimensions $(12.5, 6.25, 7.5)$. Without merging, octree decomposition alone produces an overwhelming 80.44% of blocks with cell dimensions $\mathbf{k} = (1, 1, 1)$, while 96.86% and 98.93% of blocks have dimensions at/below $(2, 2, 2)$ and $(4, 4, 4)$, respectively.

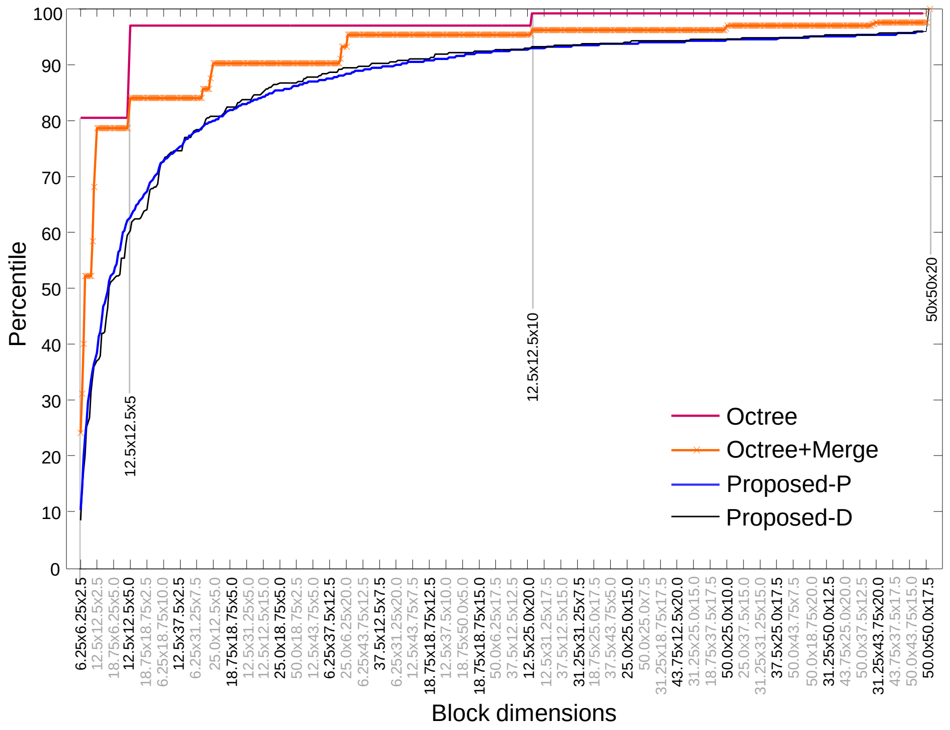


Figure 26: Cumulative distribution of block dimensions for $D = 3$: a comparison between the proposed and octree representations.

The red curve demonstrates that the blocks in the 'Octree' model are indeed highly fragmented. With intra-scale merging, the orange curve shows there is a larger spread of blocks and the proportion occupied by the smallest blocks has substantially reduced. Blocks with cell dimensions $(1, 1, 1)$, $(1, 1, 2)$, $(1, 2, 1)$, and $(2, 1, 1)$ now occupy the 24.09th, 31.26st, 39.98th, and 52.08nd pecentiles, respectively. Because this strategy does not permit inter-scale merging (outside of an octant), it cannot produce blocks with cell dimensions $(1, 1, 3)$, $(1, 3, 1)$, or $(3, 1, 1)$ and indeed $(1, 1, 4)$. These lost opportunities are capitalised by the proposed strategies. However, the 'Octree+Merge' strategy can still produce blocks of size $(1, 2, 2)$, $(2, 1, 2)$, and $(2, 2, 1)$; the so called 4-connected cells occupy the 58.45th, 68.14th, and 78.64th percentiles, respectively.

Moving up the spatial hierarchy, the $(2, 2, 2)$, $(2, 2, 4)$, $(2, 4, 2)$, and $(4, 2, 2)$ blocks correspond to the 84.01th, 85.50th, 87.36th, and 90.13th percentiles; as opposed to 96.86% at the $(2, 2, 2)$ stage for octree decomposition without merging. With intra-scale merging, the $(2, 4, 4)$, $(4, 2, 4)$, and $(4, 4, 2)$ blocks occupy the 91.07st, 92.99nd, and 95.25th percentiles. Again, blocks with intermediate cell dimensions, such as $(2, 3, 4)$ and $(3, 2, 5)$, can only be synthesized by the proposed techniques through inter-scale merging and relaxation of the dyadic constraints. Similar observations hold for more levels.

8.4 Discussion and insights

The octree comparisons with the proposed methods highlight the constraints imposed by hierarchical decomposition and the importance of block merging. In cases where the block structure is subsequently consumed by a block estimation process, minimising the block count (while maintaining the same level of spatial fidelity and domain classification granularity) can substantially reduce the computational effort associated with probabilistic inference of block attributes. The analysis has shown, for instance, when a minimum block size of $(1.5625, 1.5625, 0.625)$ and 5 decomposition levels are applied to the test site, the block count of the proposed model and 'Octree+Merge' model are 20.52% and 41.92% respectively relative to standard octree decomposition without block merging. The 'Octree+Merge' approach is probably more indicative of the flexible subblocking approach used in the *Datamine* software [24] which adjusts the split depending on the angle of intersection of a particular block with the surface controlling the sub-division. It is however unclear if a split extends all the way down within a parent block, whether different permutations and aspect ratio optimization are considered in *Datamine*. Returning to the block count comparison, the factor of 2 reduction may be attributed to inter-scale merging and relaxation of dyadic constraints as illustrated in Section 8.3. The proposed coordinate-ascent block merging approach also scales better. For instance, the block count growth rate from 3 to 5 levels is kept at 13.8152 as opposed to 16.9611 (see Table 4). From the cumulative distribution in Figure 26, we observed the block models obtained using the proposed methods are spatially less fragmented.

Although the performance metrics all point in favor of the proposed methods, the real strength of the proposal is its flexibility. First, the block merging algorithm was used on its own to consolidate and heal a spatially fragmented block model given only the parent block size and minimum block dimensions (see Section 7). Second, the same algorithm was used as part of a coherent block model spatial restructuring strategy to coalesce sub-blocks partitioned by intersecting surfaces (see Section 8). Finally, it provides a successive refinement framework for updating the spatial structure of a block model; this takes a dynamic (non-static) view and emphasizes the evolving nature of the model. In contrast, some commercial softwares [24] produce block models that cannot be altered or easily manipulated. The iterative setting implies the input blocks can have different dimensions. The standard octree paradigm is not equipped to deal with input blocks with varying sizes. As a case in point, given an input block with cell dimensions $(4, 2, 8)$, a three level dyadic decomposition would need to be applied asymmetrically to each axis, via $d_1(z), d_2(x, z), d_3(x, y, z)$, to not violate the minimum block size (lower bound and integer multiple) conditions. Furthermore, an input block with cell dimensions $(3, 5, 2)$ would leave the octree approach with no viable sub-blocking options unless the block is divided non-uniformly which would set it on a similar path as the proposed strategy notwithstanding the hierarchical embedding (rigid subblock alignment) constraints.

A crucial part of 'iterative refinement' is that large portions of an existing model should remain intact, useful information in unaffected parts of the model are not invalidated through the update process, and changes affect only localized regions that interact with new surfaces. This allows creation of a new block model from scratch given a set of surfaces (representing geological boundaries or other types of delineation), as well as automatic correction for misplaced boundaries within an existing model. In [15], the current proposal is portrayed as a system within a system, where new information captured by sampled assay data are used to manipulate the shape of relevant surfaces to maximize agreement with the latest observations. These are used in turn to update the block model to improve the accuracy of its domain structure by maximising the resolution and positional integrity of the blocks near those surfaces. This has been shown in [15] to have a positive effect on processes downstream. In particular, grade estimation performance improves as boundary (smearing) artefacts are reduced during inferencing.

8.5 Summary of contributions

The overall contributions of this paper may be summarized as follows.

1. **Developing a flexible framework** capable of altering the spatial structure of a block model given boundary constraints in the form of triangle mesh surfaces and a set of tagging instructions.
2. **Describing computational techniques** for all system components
    - From block-surface intersection detection (identifying areas requiring an update), block structure decomposition (improving boundary localization), block consolidation (increasing compactness of representation) to block tagging (encoding the location of blocks relative to the given surfaces).
3. **Devising a new algorithm** (coordinate-ascent inspired block merging) that is amenable to multi-threading.
4. **Demonstrating real applications**: The algorithm was used firstly to reduce block fragmentation in an existing mine resource estimation block model where parameters are fixed and no surfaces are given; secondly as an integrated component within the block model creation/augmentation workflow to coalesce subblocks partitioned by intersecting surfaces.
5. **Providing insights** through quantitative analysis, highlighting the constraints of dyadic hierarchical decomposition and importance of inter-scale merging for achieving a compact block model; discussing why this matters for subsequent block-based estimation processes (see Section 8.1–8.4).

The first point includes correcting the position of previously misplaced boundaries in an existing model (see Figure 19). Philosophically, it challenges the perception that voxel-based models are hard to modify once they have been constructed. It demonstrates that local refinement (including consolidation of fragmented blocks from previous updates) is both possible and can be done efficiently in an iterative setting; with changes to the model confined to areas that interact with new surfaces. The second point is about greater transparency. In 3D geological modeling papers, models are often generated using commercial software where the fundamental techniques employed are unclear. This paper sheds light on this and discusses pit falls and robust solutions from an engineering perspective. The third point emphasizes flexibility. Depending on user requirement on whether internal subblock boundaries are to be respected, it can operate in two modes (see Figure 18): with *persistent block memory* to ensure an input block always maps uniquely to a merged output block, or with *dissolved subblock boundaries* to achieve higher block compaction (a lower block count). The specific objective in the fourth point was to capture geological structures of an ore body in a block model at a specified resolution with a minimal block count. However, the method can potentially be applied elsewhere.

9 Conclusions

This paper described a framework for updating the spatial structure of a 3D geology block model given mesh surfaces. The system consists of four components which perform block-surface overlap detection, spatial structure decomposition, sub-blocks consolidation and block tagging, respectively. These processes are responsible for identifying areas where refinement is needed, increasing spatial resolution to minimize surface approximation error, reducing redundancy to increase the compactness of the model and establishing the domain of each block with respect to geological boundaries. A flexible architecture was presented which allows a model to be updated simultaneously, or iteratively, by multiple surfaces, to selectively retain or modify existing block domain labels. Robustness and accuracy of the system were considered during the design; one notable feature was using ray-tracing to establish the location of sub-blocks relative to surfaces, particularly those near boundaries, prior to block consolidation to minimize boundary distortion. Other techniques employed include block-surface intersection analysis based on the separable axis theorem.

The main contribution was a coordinate-ascent merging algorithm which is used during block consolidation in the proposed framework. This technique was extended to solve a related problem, viz., using block merging to reduce fragmentation in an existing block model where surfaces are not involved. Issues relating to scan pattern, merging conventions were discussed, differences between *'persistent block memory'* and *'dissolved sub-block boundaries'* were explained in terms of internal dimensions compatibility and sub-block alignment constraints imposed on the expansion feasibility test. Performance was evaluated with respect to block aspect ratio, output block count and processing time using a multi-threaded implementation. The results demonstrated the quality and scalability of the proposed technique. Systematic evaluation against octree subblocking further highlights the limitations of hierarchical decomposition and the importance of inter-scale merging and relaxation of dyadic constraints. Algorithm 3 produced merged blocks with less extreme aspect ratios and the approach is well suited to parallel processing. The techniques described may apply to areas outside of geoscience (see Figure 17) where 3D body localization, also known as 3D region segmentation, is required inside a block model given some mesh surfaces.

10 Appendices

The appendices are contained in the supplementary material:

- (A) **Method for block triangle overlap detection** [1];
- (B) **Side-of-surface determination via ray tracing** [6] [21];
- (C) **Demonstration on the Stanford Bunny**; data courtesy of [19], [30];
- (D) **Pseudocode**;
  - Algorithm 1: Coordinate-ascent merging algorithm (as used for spatial restructuring in Section 2.3)
  - Subroutine 1: Compute sub-block properties
  - Algorithm 2: Coordinate-ascent merging algorithm (for model de-fragmentation in Appendix D)
  - Subroutine 2: Feasibility tests and state updates during block expansion
  - Algorithm 3: Block merging with multiple scans and optimized block aspect ratios
- (E) **Octree decomposition and merging**;
- (F) **Detailed octree subblocking comparison**.

Acknowledgment

This work was supported by the Australian Centre for Field Robotics and the Rio Tinto Centre for Mine Automation. The author is grateful to the anonymous reviewers whose comments and perspectives have helped improved this paper. Dr. Gregory Cresp and Dr. Alexander Lowe are thanked for providing feedback and critically evaluating the system. Dr. Tamara Vasey is thanked for reviewing this manuscript.

References

[1] AKENINE-MÖLLER, T. Fast 3D triangle-box overlap testing. *Journal of graphics tools 6*, 1 (2001), 29–33. doi:10.1080/10867651.2001.10487535.

[2] ARVIDSSON, M., AND GREMYR, I. Principles of robust design methodology. *Quality and Reliability Engineering International 24*, 1 (2008), 23–35. doi:10.1002/qre.864.

[3] BALL, A., SILVERSIDES, K., CHLINGARYAN, A., AND MELKUMYAN, A. Creating large scale probabilistic boundaries using Gaussian Processes. In *Proceedings of Geostats 2020* (Toronto, Canada, 2021), International Geostatistics Congress.

[4] CASTAGNAC, C., TRUFFERT, C., BOURGINE, B., AND COURRIOUX, G. French Geological Survey (Bureau de Recherches Géologiques et Minières): Multiple software packages for addressing geological complexities. In *Synopsis of Current Three-dimensional Geological Mapping and Modeling in Geological Survey Organizations*, R. C. Berg, Ed., vol. 578. Illinois State Geological Survey Circular, 2011, ch. 8, pp. 42–47.

[5] CLOUT, J. Iron formation-hosted iron ores in the Hamersley Province of Western Australia. *Applied Earth Science 115*, 4 (2006), 115–125. doi:10.1179/174327506X138931.

[6] DIETRICH, A., GOBBETTI, E., AND YOON, S.-E. Massive-model rendering techniques: a tutorial. *IEEE Computer Graphics and Applications 27*, 6 (2007), 1–18. doi:10.1109/MCG.2007.154.

[7] DIMITRAKOPOULOS, R., AND RAMAZAN, S. Stochastic integer programming for optimising long term production schedules of open pit mines: methods, application and value of stochastic solutions. *Mining Technology 117*, 4 (2008), 155–160. doi:10.1179/174328609X417279.

[8] DRAGIEV, S., TOUSSAINT, M., AND GIENGER, M. Gaussian process implicit surfaces for shape estimation and grasping. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), IEEE, pp. 2845–2850. doi:10.1109/ICRA.2011.5980395.

[9] EMERY, X. Simple and ordinary multigaussian kriging for estimating recoverable reserves. *Mathematical Geology 37*, 3 (2005), 295–319. doi:10.1007/s11004-005-1560-6.

[10] FELTRIN, L., MCLELLAN, J., AND OLIVER, N. Modelling the giant, Zn–Pb–Ag Century deposit, Queensland, Australia. *Computers & Geosciences 35*, 1 (2009), 108–133. doi:10.1016/j.cageo.2007.09.002.

[11] JONES, C. B. Data structures for three-dimensional spatial information systems in geology. *International Journal of Geographical Information System 3*, 1 (1989), 15–31. doi:10.1080/02693798908941493.

[12] LEAPFROG. Leapfrog Geo: Trainagulated Meshes. https://help.leapfrog3d.com/Geo/4.3/en-GB/Content/meshes/triangulated-meshes.htm, 2019. Accessed: 2019-10-25.

[13] LEITE, A., AND DIMITRAKOPOULOS, R. Stochastic optimisation model for open pit mine planning: application and risk analysis at copper deposit. *Mining Technology 116*, 3 (2007), 109–118. doi:10.1179/174328607X228848.

[14] LEUNG, R. Subsurface boundary geometry modeling: Applying computational physics, computer vision, and signal processing techniques to geoscience. *IEEE Access 7* (2019), 161680–161696. doi:10.1109/access.2019.2951605.

[15] LEUNG, R., LOWE, A., CHLINGARYAN, A., MELKUMYAN, A., AND ZIGMAN, J. Bayesian surface warping approach for rectifying geological boundaries using displacement likelihood and evidence from geochemical assays. *arXiv e-prints* (2020), arXiv:2005.14427.

[16] LI, N., BAGAS, L., LINDSAY, M., WEDGE, D., BAI, L., AND SONG, X. An irregular triangle mesh buffer analysis method for boundary representation geological object in three-dimension. *Earth Science Informatics 10*, 2 (2017), 149–167. doi:10.1007/s12145-016-0283-1.

[17] LUO, Z. Q., XIAO-MING, L., JIA-HONG, S., YA-BIN, W., AND WANG-PING, L. Deposit 3D modeling and application. *Journal of Central South University of Technology 14*, 2 (2007), 225–229. doi:10.1007/s11771-007-0045-9.

[18] MATHERS, S. J., KESSLER, H., AND NAPIER, B. British Geological Survey: A nationwide commitment to 3-D geological modeling. In *Synopsis of Current Three-dimensional Geological Mapping and Modeling in Geological Survey Organizations*, R. C. Berg, Ed., vol. 578. 1Illinois State Geological Survey Circular, 2011, ch. 6, pp. 25–30.

[19] MCGUIRE, M. Computer Graphics Archive. https://casual-effects.com/data/, July 2017.

[20] MIRA GEOSCIENCE. 3D Geological Modelling. http://webcache.googleusercontent.com/search?q=cache:http://www.mirageoscience.com/our-products/software-solutions/3d-geological-modelling, 2018. Accessed: 2018-12-23.

[21] MÖLLER, T., AND TRUMBORE, B. Fast, minimum storage ray-triangle intersection. *Journal of graphics tools 2*, 1 (1997), 21–28. doi:10.1080/10867651.1997.10487468.

[22] NEWMAN, T. S., AND YI, H. A survey of the marching cubes algorithm. *Computers & Graphics 30*, 5 (2006), 854–879. doi:10.1016/j.cag.2006.07.021.

[23] PHARR, M., JAKOB, W., AND HUMPHREYS, G. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[24] PONIEWIERSKI, J. Block model knowledge for mining engineers – an introduction. https://www.deswik.com/wp-content/uploads/2019/07/Block-model-knowledge-for-mining-engineers-An-introduction.pdf, 2019. Accessed: 2020-08-08.

[25] ROSSON, M. B., AND CARROLL, J. M. Scenario based design. In *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, J. Jacko and A. Sears, Eds. CRC Press, 2002, ch. 53, pp. 1032–1050.

[26] SAMET, H. An overview of quadtrees, octrees, and related hierarchical data structures. In *Theoretical Foundations of Computer Graphics and CAD*. Springer, 1988, pp. 51–68. doi:10.1007/978-3-642-83539-1_2.

[27] SEDLMAIR, M., MEYER, M., AND MUNZNER, T. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2431–2440. doi:10.1109/TVCG.2012.213.

[28] SRK CONSULTING. Block Modelling ("3D Modeling Project, Spout Lake, British Columbia, Canada"). http://webcache.googleusercontent.com/search?q=cache:https://www.srk.ru.com/en/block-modelling, 2018. Accessed: 2018-12-25.

[29] THORNE, W., HAGEMANN, S., WEBB, A., AND CLOUT, J. Banded iron formation-related iron ore deposits of the Hamersley Province, Western Australia. In *Banded Iron Formation-Related High-Grade Iron Ore*. Society of Economic Geologists, 2008, pp. 197–221.

[30] TURK, G., AND LEVOY, M. The Stanford 3D Scanning Repository, August 2014.

[31] VASUDEVAN, S., RAMOS, F., NETTLETON, E., AND DURRANT-WHYTE, H. Gaussian process modeling of large-scale terrain. *Journal of Field Robotics 26*, 10 (2009), 812–840. doi:10.1002/rob.20309.

[32] WANG, G., LI, R., CARRANZA, E. J. M., ZHANG, S., YAN, C., ZHU, Y., QU, J., HONG, D., SONG, Y., HAN, J., MA, Z., ZHANG, H., AND YANG, F. 3D geological modeling for prediction of subsurface Mo targets in the Luanchuan district, China. *Ore Geology Reviews 71* (2015), 592–610. doi:10.1016/j.oregeorev.2015.03.002.

[33] WANG, G., ZHANG, S., YAN, C., SONG, Y., SUN, Y., LI, D., AND XU, F. Mineral potential targeting and resource assessment based on 3D geological modeling in Luanchuan region, China. *Computers & Geosciences 37*, 12 (2011), 1976–1988. doi:10.1016/j.cageo.2011.05.007.

[34] WYCISK, P., HUBERT, T., GOSSEL, W., AND NEUMANN, C. High-resolution 3D spatial modelling of complex geological structures for an environmental risk assessment of abundant mining and industrial megasites. *Computers & Geosciences 35*, 1 (2009), 165–182. doi:10.1016/j.cageo.2007.09.001.