Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 12, No. 04, December 2026.

MEDIAN SPLIT AGAINST THE SURFACE AREA HEURISTIC FOR BROAD-PHASE COLLISION DETECTION UNDER PER-FRAME REBUILD: A PAIRED AND DIRECTLY TIMED COMPARISON

Hamzah Asyrani Sulaiman[1], Che Ku Nuraini Che Ku Mohd[1], Mohamad Lutfi Dolhalit[1], Abdullah Bade[2] and Iznora Aini Zolkifly[3]

(Received: 24-Aug.-2026, Revised: 5-Oct.-2026 and 14-Oct.-2026, Accepted: 15-Oct.-2026)

ABSTRACT

For broad-phase hierarchies rebuilt every frame, the submitted three-axis surface area heuristic performs 17.8 to 46.0 percent fewer box tests than the object median yet takes 9 to 365 percent more net time throughout, up to 2,000 objects; no crossover beyond is excluded. So does a one-axis heuristic. With a median split at eight objects or fewer, it led by 1.1 to 2.4 percent in 12 configurations, 11 when repeated and 10 in a fourth session, too small to claim. A split at the midpoint of the node's box was faster than the object median in all 30 density-controlled configurations, by 21 to 39 percent as implemented. A hierarchy built in Morton order, measured at a referee's request, was faster than both at 100 objects or more, mainly in construction. No heuristic was faster than the midpoint split, nor the variant than the Morton build.

KEYWORDS

Broad-phase collision detection, Bounding volume hierarchy, Surface area heuristic, Median split, Per-frame rebuild.

1. INTRODUCTION

A simulation of N moving bodies presents $N(N-1)/2$ distinct pairs in every frame and only a small fraction of them are close enough to interact. The broad-phase is the stage that discards the rest. It encloses every body in a simple proxy volume, here an axis-aligned box and reports the pairs the proxies of which overlap; the field is surveyed by Weller [1]. One family of broad-phase methods groups the proxies into a bounding volume hierarchy, from one volume per object up to one for the whole scene and finds the overlapping pairs by testing the tree against itself [2].

A hierarchy built from the root downwards must decide at every node how to divide its objects between two children. An inexpensive rule divides them at the median on the longest axis. The surface area heuristic instead estimates the expected traversal cost of each candidate division and selects the cheapest [3]; it costs more to evaluate and is meant to return that cost through a cheaper traversal. Whether the exchange pays has been measured in two settings. For ray tracing, Wald [3] rebuilt every structure in every frame and tabulated construction time beside rendering performance for three builders. For collision detection, Klosowski et al. [4], Ytterlid and Shellshear [5] compared splitting rules and reported construction time beside query time, for hierarchies over triangles that are built once, before the simulation begins.

A broad-phase that rebuilds its hierarchy in every frame is a third setting and there, in the published sources examined for this study, the answer has been asserted and not measured. Karras [6] observes that such a structure is reconstructed at every time step and then answers a fairly small number of queries and concludes that construction speed is of primary importance. Wang et al. [7] record the same preference of collision detection for construction speed over tree quality. That preference is a design position unaccompanied by a measurement. The present paper supplies the measurement for one pair of rules, the object median and the binned surface area heuristic of the submitted version. It reports construction time, traversal work and net time, the last being the whole cost of a frame, on paired and

1. H. A. Sulaiman, C. K. N. C. K. Mohd and M. L. Dolhalit are with Uni. Teknikal Malaysia Melaka, Malaysia. Emails: {asyrani, cknuraini, lutfi}@utem.edu.my
2. A. Bade is with Uni. Malaysia Sabah, Jalan UMS, Sabah, Malaysia. Email: abb@ums.edu.my
3. I. A. Zolkifly is with Uni. Tenaga Nasional (UNITEN) Putrajaya Campus, Selangor, Malaysia. Email: iznora.aini@uniten.edu.my

seeded scenes in which the object count and the packing density are varied independently and it locates or bounds the object count at which the heuristic repays its construction. No novelty is claimed beyond that measurement.

The paper makes three contributions: a count of the work of construction under both rules, derived from the code before any measurement, which corrects the model of the submitted version; the measurement itself, made under a design, criteria and an analysis registered in advance, in three forms of implementation and with sweep and prune and a uniform grid as baselines from the other families; and a correction of the record: the principal metric of the submitted version counted only the box tests that succeeded, its ratios against brute force and its narrow-phase recommendation are withdrawn and every measurement reported here is new.

The measurement returns a negative answer for the heuristic as it was submitted, which searches twelve bins on three axes at every node. Over the 30 configurations that cross object count with density, it performs between 17.8 and 46.0 percent fewer box tests than the median split and in net time the median split is the faster in every configuration and in every form of implementation, with no crossover up to 2,000 objects and none excluded beyond. Withholding the search from nodes of 4 or 8 objects and fewer narrows the difference and does not reverse it. For this rule, the result agrees with the position of Karras and of Wang et al. A second campaign, registered after the first had been read, tested a one-axis heuristic. Run at every node, it is the slower everywhere as well; with the median split at nodes of 8 objects or fewer it is the faster, by at most 2.4 percent, in 12 of the 30 configurations. A repeat session reproduced the label in 11 of the 12; the margin is not presented as a practical advantage. A third campaign then set a split at the midpoint of the node's box beside the object median: as implemented, it was the faster in every configuration and no heuristic tested was faster than it.

2. Related Work

Broad-phase methods are commonly grouped into three families. Sorting methods, of which sweep and prune is the best known, sort the projected extents of the boxes on the coordinate axes and re-sort them incrementally from frame to frame [8]-[9]. Spatial sub-division divides space and not the objects and has been combined with sweep and prune to handle objects of non-uniform size [10]. Hierarchical methods divide the objects. Serpa and Rodrigues [2] place many broad-phase algorithms on a common footing in Broadmark, the three scenes of which rank the algorithms differently. They hold the density of objects fixed as the object count grows and they report one time per algorithm in which every step of the broad-phase is included, so that construction is not shown apart from the query. Lubbe et al. [11] do report construction and traversal as separate shares of a simulation step and Sung and Hong [12] vary the broad-phase with the narrow-phase held fixed. In none of these three benchmarks is the split rule of a hierarchy a factor.

The comparison of split rules for collision hierarchies is nevertheless old. Klosowski et al. [4] compared four rules for the split axis and two split points, the mean and the median of the centroids, for a hierarchy over the triangles of an environment and printed construction time beside query time. They found that an inexpensive axis rule was built an order of magnitude faster than the rule that minimizes the summed volume of the two children while its queries were nearly as fast and that the median never gave faster queries than the mean. van den Bergen [13] reported the midpoint of the box ahead of a division into two sets of equal size. Zachmann [14] derived, from the probability that two child boxes overlap during a simultaneous traversal, the criterion that a split should minimize the summed volume of the children. Ytterlid and Shellshear [5] set a mean split against splits chosen by several cost functions, the surface area among them and tabulated build time beside query rate; a later review lists the same strategies and split points [15]. All of these hierarchies are built once, over triangles, for a query of one tree against another, so that construction is preprocessing and no net time exists. Two consequences for the present study are declared here. The criterion that this literature derives for a traversal of tree against tree is the volume of the children and not their surface area; the surface area form is examined, because it is the rule that the submitted version tested. And the median examined in the first two campaigns is the object median, which Klosowski et al. found and van den Bergen reported weaker than the mean and the midpoint.

The surface area heuristic comes from ray tracing, where MacDonald and Booth [16] studied it for space

sub-division on the intuition that surface area estimates the probability that a ray meets a volume. Wald [3] carried its binned approximation over to hierarchies and, rebuilding every structure in every frame, tabulated build time beside rendering performance. In his experiments, a build that splits at the spatial median, which there halves a cell of a recursive sub-division of the scene and is neither the object median nor the midpoint split measured here, delivered between 76 and 87 percent of the traversal performance of a full sweep build on five models. For rays, a comparison of the two medians does exist: Wald, Boulos and Shirley [17] tabulate a spatial median build ahead of an object median build in rendering rate on three static scenes, without defining either build. Later builders pursue the same tension: Lauterbach et al. [18] reduced construction to a sort along a Morton curve, accepting a poorer tree for a much faster build, Karras and Aila [19] restructure such a tree to recover its quality, PLOC++ [20] approaches the quality of a binned build at close to that speed and H-PLOC [21] reaches similar quality in a single-kernel launch. The survey of Meister et al. [22] records that trees forced to split at the object median are of lower quality than trees optimized by the heuristic and hierarchies of tighter volumes have likewise been developed for ray tracing [23]-[25]. All of this work concerns ray tracing.

For a broad-phase the conclusion has, in the published sources examined, been stated without being measured: the position of Karras [6] and of Wang et al. [7] was given in Section 1 and Serpa and Rodrigues [2] describe hierarchies on the processor as usually built from the top down at the median point of the objects or incrementally. Chitalu et al. [26] report that rebuilding their hierarchy in every frame is faster than refitting it. A Morton build has been measured on the processor against a surface area heuristic build for a broad-phase over the proxies of triangle meshes, built once per frame [27]; the Morton build here is a separate implementation, measured in the regime of this paper. Mandarapu et al. [28], who find colliding spheres by ray queries on ray tracing hardware, sweep the interval between rebuilds and name the trade between the cost of maintaining a hierarchy and the quality of its traversal. Much other recent work on collision and proximity queries concerns the query and the hardware it runs on: detection by ray tracing over a bounding mesh [29], distance computation on the graphics processor [30], continuous detection against signed distance fields [31], the Gilbert-Johnson-Keerthi procedure in discrete element simulation [32] and traversal over distributed meshes [33]. Wang et al. [34] benchmark continuous detection against an exactly computed ground truth and report missed collisions beside running time, which is why every engine here is checked against an exhaustive reference before its time is interpreted. To our knowledge, no published study compares an object median with a binned surface area heuristic for a broad-phase that is rebuilt in every frame while reporting construction time, traversal work and net time together. That comparison is the subject of this paper.

3. CONSTRUCTION RULES AND COST MODEL

This section states both rules as the code executes them and derives from them a count of the work of construction. The count was written down before any measurement was taken and it replaces the model of the submitted version, the errors of which are set out at the end of Sub-section 3.2.

3.1 The Two Rules As Implemented

Both rules build a binary tree from the root downwards over the N axis-aligned boxes of a frame. The box of a node is the union of the boxes of its objects and a node is a leaf if and only if it holds exactly one object; neither rule has any other stopping condition. Every tree is therefore a full binary tree with N leaves and $N - 1$ internal nodes, whatever its balance. Because every leaf holds one object under both rules, the set of candidate pairs that a correct traversal hands downstream is the same for both rules on a given scene and the rules can differ only in the work done to produce it. The choice of rule consequently cannot change the workload of a narrow-phase and the recommendation that the submitted version made on that ground is withdrawn. The median split selects the axis on which the box of the node is longest, sorts the objects by the centre of their boxes on that axis and assigns the first $\lfloor m/2 \rfloor$ of the m objects to the left child and the remainder to the right, so that the shape of its tree depends on N alone.

The heuristic rule evaluates the surface area heuristic over bins, the approximation that Wald [3] applied to the construction of hierarchies. Wald bins on the single axis on which the centres are most spread, whereas the rule examined here bins on all three axes. On each axis, the centres of the boxes of the node span an interval $[c_{\min}, c_{\max}]$ and an axis on which that interval is narrower than 10^{-6} units is skipped.

On every other axis, an object the centre of which is at c is assigned to one of $K = 12$ bins of equal width,

$$b = \min\left(K - 1, \left\lfloor K \frac{c - c_{\min}}{c_{\max} - c_{\min}} \right\rfloor\right), \tag{1}$$

and each bin accumulates the number of its objects and the union of their boxes. A forward and a backward sweep over the bins accumulate the box and the object count on either side of each of the $K - 1$ boundaries between bins, at a cost of $2(K - 1) = 22$ box merges per axis. A boundary that leaves either side empty is not evaluated. Every other boundary is a candidate division, of cost

$$C = C_{\text{trav}} + \frac{A_L}{A_P} n_L + \frac{A_R}{A_P} n_R, \tag{2}$$

where A_P, A_L and A_R are the surface areas of the box of the node and of the boxes of the two sides, n_L and n_R are the object counts of the two sides, the cost of testing one object is the unit and $C_{\text{trav}} = 1$. The candidate of lowest cost over all axes is kept, the first one found being retained when two are equal and the objects are divided at the chosen boundary by repeating the bin assignment on the chosen axis. The number of bins and the traversal constant are those of the submitted version in every form examined.

Two conditions replace the heuristic division by a median split and neither of them stops the recursion. The first is the leaf-cost check. The cost of leaving a node of m objects undivided is taken to be m and if no candidate was evaluated or if the best candidate costs at least m, the node is divided at the median of its longest axis exactly as the median rule would divide it. The submitted version described this check as preventing a split. That description was incorrect: the check changes which division is made and never whether one is made. The second condition guards against a degenerate division: if dividing at the chosen boundary leaves one side empty, the node is divided at the median of the chosen axis instead.

3.2 Count of Construction Work

Construction work is counted in three kinds of operation that are not assumed to cost the same: an object pass, which is one visit to one object inside one loop over the objects of a node; a comparison inside the sort; and fixed work, the sweep merges and candidate evaluations, the number of which at a node does not depend on its size. Let D denote the sum of the depths of the leaves, which equals the sum of the sizes of the internal nodes. The median split makes two passes over the objects of every internal node, one to form its box and one to copy its objects into the two child lists and one pass at every leaf, so that it performs exactly $N + 2D$ object passes, where $D = s(N)$ with $s(1) = 0$ and $s(m) = m + s(\lfloor m/2 \rfloor) + s(\lceil m/2 \rceil)$. The number of comparisons is a property of the library sort and is not predicted; it is reported against the sum of $m\log_2 m$ over the internal nodes. The heuristic makes nine passes over the objects of an internal node: one to form its box, one on each axis to bound the centres, one on each axis to assign the bins and two to divide the objects at the chosen boundary. A skipped axis saves one pass, a node that takes the leaf-cost fallback makes one pass fewer and a node that takes the degenerate fallback makes one more. The exact number of object passes is therefore

$$P_{\text{SAH}} = N + 9D - S_{\text{skip}} - S_{\text{cost}} + S_{\text{deg}}, \tag{3}$$

where $S_{\text{skip}}, S_{\text{cost}}$ and S_{deg} are the node sizes summed over the skipped axes, over the nodes that took the leaf-cost fallback and over the nodes that took the degenerate fallback. For a balanced tree with no skipped axis and no fallback, the count is $N + 9s(N)$, which is a reference value and not a bound, since an unbalanced tree has a larger D. The fixed work is at most 66 sweep merges and 33 candidate evaluations at each of the $N - 1$ internal nodes, 99 operations per node and it is smaller where an axis is skipped or a boundary has an empty side. A leaf returns before any search and incurs none of it.

The submitted version modelled the heuristic as $3N\log_2 N + (2N - 1)(72 + 33)$ operations and that model was wrong in three respects. It charged 105 fixed operations to all $2N - 1$ nodes, although a leaf performs none and an internal node at most 99, which overstated the fixed term by a factor of at least 2.12. It counted three passes over each object at each level where the code makes nine and it took the number of levels to be $\log_2 N$, which holds only for a balanced tree, whereas the term scales with D. And it added passes, comparisons and fixed work into one total as though each cost the same.

Counts of different kinds are accordingly not summed into a time. Solely so that the two models can be

compared, the ratio of heuristic to median work under an equal weighting of every operation is 4.76 at $N = 50$ and 2.24 at $N = 2{,}000$ in the corrected model for a balanced tree, where the submitted model gave 11.99 and 3.70; the table of reference values is in the archive and the equal-weight ratio is not a prediction of the ratio of construction times. About time, the model predicts a direction only; namely, that construction by the heuristic takes longer at every object count. A ratio that narrows as N grows is what the model suggests, but a fixed cost per node common to both rules would produce the same, so an observed narrowing could fail to refute the model and could not by itself confirm it. The flat form of Sub-section 4.1 applies the same rules with fewer passes and with a selection in place of a sort; its counts are reported and no identity is claimed for them.

4. METHODOLOGY

4.1 Engines and Forms

The answer to a question about time can depend on how a hierarchy is implemented as much as on its split rule, so each rule is run in three forms, listed with the baselines in Table 1. Form S is the code of the submitted version, copied without algorithmic change, in which the median split engine was called StaticBVH and the heuristic engine DynamicBVH; because both are rebuilt in every frame, those names are not used again. Besides its symmetric traversal, form S carries a difference between the two engines that is not the split rule: the median engine releases the tree of the previous frame after building the new one and the heuristic engine before, so that the two builds do not present the same heap to the allocator. Form C keeps the construction code and removes both defects, with a canonical traversal and a release of the previous tree in a separately timed stage that is identical for both rules. Form F applies the same two rules over arrays that are reused from frame to frame, with boxes and centres computed once per frame and the median found by selection. Its split decisions are evaluated on the same floating-point values as in the other forms, so forms C and F build the same tree wherever no two centres are equal at a median boundary. Form F is a plain flat build and not a tuned one, without explicit vectorization, wide nodes or refitting. The conclusions about time are drawn from form F and forms S and C are reported beside it.

The submitted heuristic runs its full search at every internal node, down to a node of two objects, where the search cannot change the outcome and still incurs the fixed work of Sub-section 3.2. Two further engines in form F therefore apply the heuristic only at nodes of more than 4 or more than 8 objects and the median split at every smaller node. Both are run in every cell, so that nothing is tuned. They form a secondary comparison, declared as such before the data were collected: the question of net time is answered on the submitted rule and the cut-off engines show only whether that answer depends on the search at small nodes.

The remaining engines position the hierarchies. BF tests every unordered pair of objects, rebuilding both boxes for every pair as submitted and BF-F does so on boxes computed once per frame. SAP stands for the sorting family, of which sweep and prune [8] is the best-known member and GRID for the spatial sub-division family [35]. The grid emits a pair only from the cell that contains the minimum corner of the intersection of its two boxes, which removes the need for a container to eliminate duplicates and its three cell sizes are all run in every cell, so that none is tuned. Both baselines are rebuilt from nothing in every frame and keep no state between frames, so they are simpler than the incremental methods of the cited sources and represent their families only on those terms. Rebuilding everything in every frame is the regime under study. It is a choice of this study and not a requirement of physics engines and refitting and incremental update of a hierarchy [36] lie outside its scope. Every engine applies the same overlap predicate, under which boxes that touch count as overlapping.

MED denotes the median split and SAH the surface area heuristic; the suffix names the form. Figure 1 gives the two traversals. Both rest on the procedure Visit, which applies one box test to a pair of nodes and, if the boxes overlap, either emits the pair of objects when both nodes are leaves or descends one node. The descent rule is that of the submitted version in every form: an internal node is descended in preference to a leaf and of two internal nodes the one of larger volume, the first operand being taken when the volumes are equal. The symmetric traversal starts Visit on the root paired with itself. It consequently tests every node against itself, each child against its parent and both orderings of every pair of distinct nodes that it reaches; a leaf paired with itself emits nothing and every overlapping pair of objects is emitted twice and reduced to one by an ordered set keyed on the pair with its smaller index

first. The canonical traversal calls Visit once on the two children of every internal node. Its operands always lie in disjoint sub-trees, so no node is tested against itself, each unordered pair of distinct nodes that the descent reaches is tested once, each overlapping pair of objects is emitted once and no deduplication is required.

Table 1. The fourteen engines of the first campaign, the two added in the second and the one added in the third, all of which are rebuilt in every frame.

Engine	Family	Construction	Query and duplicates	Role
BF	Exhaustive	None; both boxes rebuilt for every pair	Every unordered pair of objects	Baseline as submitted; reference pair set
BF-F	Exhaustive	Boxes computed once per frame	Every unordered pair of objects	Baseline on equal terms with form F
MED-S, SAH-S	Hierarchy, form S	One heap node per tree node; index list copied at every node; previous tree released inside construction	Symmetric; duplicates removed by an ordered set	As submitted
MED-C, SAH-C	Hierarchy, form C	As form S; previous tree released in a separate stage	Canonical; no duplicates arise	Control for the traversal and the release
MED-F, SAH-F	Hierarchy, form F	Node array; one index array divided in place; median by selection	Canonical; no duplicates arise	Form from which conclusions about time are drawn
SAH4-F, SAH8-F	Form F with cut-off	Heuristic at nodes of more than 4 or 8 objects, median split at smaller nodes	Canonical; no duplicates arise	Secondary comparison
SAH1-F, SAH1-8-F	Form F, one axis	Heuristic binned on the axis of greatest centre extent; the second with the median split at nodes of 8 objects or fewer	Canonical; no duplicates arise	Second campaign
MID-F	Form F, midpoint	Node divided at the midpoint of its box on its longest axis or at the object median where a side would be empty	Canonical; no duplicates arise	Third campaign
SAP	Sorting	Boxes sorted by lower bound on the axis of greatest centre variance	One sweep of that axis	Family baseline
GRID-1.0, GRID-1.5, GRID-2.0	Uniform grid	Counting sort into every cell a box touches; cell edge 1.0, 1.5 or 2.0 times the largest box edge of the frame	Pairs within each cell; minimum-corner ownership	Family baseline

```
procedure VISIT(a, b, out)
  if box(a) and box(b) do not overlap then return
  if a and b are both leaves then
    if their objects i and j differ then
      append (min(i, j), max(i, j)) to out
    return
  if a is internal and (b is a leaf or
    volume(a) ≥ volume(b)) then
      VISIT(left(a), b, out); VISIT(right(a), b, out)
  else
      VISIT(a, left(b), out); VISIT(a, right(b), out)
```

```
procedure SYMMETRIC(root)                    form S
  raw ← empty list; VISIT(root, root, raw)
  seen ← empty ordered set; out ← empty list
  for each pair in raw do
    if the pair is not in seen then
      insert it in seen; append it to out
  return out
procedure CANONICAL(n, out)                  forms C and F
  if n is a leaf then return
  CANONICAL(left(n), out)
  CANONICAL(right(n), out)
  VISIT(left(n), right(n), out)
```

Figure 1. The traversal procedures. Visit is common to every form and performs one box test per call. Symmetric is the traversal of form S, as submitted. Canonical is the traversal of forms C and F and is called on the root with an empty list.

Let T be the number of box tests that the canonical traversal performs on a given tree and B_S the number that the symmetric traversal performs on the same tree. The symmetric traversal performs three tests at each of the $N - 1$ internal nodes that the canonical one does not; namely the node against itself and each of its two children against it, together with one test of each of the N leaves against itself and it performs twice every test that the canonical traversal performs once. Hence

$$B_S = 2T + 4N - 3. \tag{4}$$

The identity holds exactly whenever no two distinct internal nodes that are compared have equal volume. Where they have, the two orderings of a pair may descend different nodes; such frames are counted and reported and are not held to the identity. The form of the traversal therefore cannot change a comparison of counts between the two rules, but it roughly doubles the work of traversal relative to that of construction and so can change a comparison of times. Since the heuristic buys a cheaper traversal with a dearer construction, a doubled traversal is expected to favour it and forms C and F were added, so that the difference between forms could be measured and not assumed.

4.2 Scene Arms

Bodies are cubes of half extent h. They do not respond to one another: each moves at a velocity the components of which are drawn uniformly from [-3, 3] units per second, is constant between reflections and is reversed at the wall of a cubic world of half extent L, so that its box always lies inside the world and the trajectory of a scene does not depend on the engine being measured. Density is expressed as the packing fraction $\phi = NE[V]/(2L)^3$, where $E[V]$ is the mean volume of a body, equal to 2.16 cubic units when h is uniform on $[0.3, 0.9]$. Because the bodies pass through one another, the dense cells contain configurations that an engine with contact response would not sustain, a limitation taken up in Section 7. The five arms of Table 2 are run.

Table 2. The scene arms. Lengths are in world units and velocities in units per second.

Arm	Scene	Packing fraction ϕ	N	Cells
A, replication	Centres uniform on $[-6,6]^3$; wall at ± 8; h uniform on [0.3, 0.9]	Not controlled; $NE[V]/16^3$ is 0.026 to 0.264	50, 100, 200 and 500	4
B, density controlled	Each centre uniform on $[-(L-h), L-h]$ on every axis; h uniform on [0.3, 0.9]; L set from ϕ	1/32, 1/16, 1/8, 1/4 and 1/2	50, 100, 200, 500, 1,000 and 2,000	30
C, clustered	Clusters of 25 bodies. A cluster is a box of half extent 3.00, packed to 1/4 inside, that moves and reflects in the world as a body does; each body moves and reflects inside its cluster and its velocity is the sum of the two	1/32 over the world	as B	6
H, heterogeneous sizes	As B, with h log-uniform on [0.15,1.8], for which $E[V] = 6.255$	1/8	as B	6
R, loose proxies	As B, each cube rotating at 0.5 to 2.0 rad/s about a uniformly random axis; centres uniform on $[-(L-\sqrt{3}h), L-\sqrt{3}h]$	1/8 of body volume	as B	6

Arm A reproduces the generator of the submitted version, in which the density rises with N and drifts within a run as the bodies spread from an inner cube towards the wall. Arm B crosses the object count with the packing fraction, which the submitted design confounded and draws every centre, so that the distribution of positions is stationary. Arms C, H and R each change one parameter of arm B: the bodies are gathered into moving clusters, their sizes are spread over a ratio of 12 to 1 or they rotate, so that the proxy, which is the axis-aligned box enclosing the rotated cube, fits loosely and changes from frame to frame. That proxy is recomputed in the scene step, outside every timed span, from initial orientations that are uniform random rotations [37].

To a broad-phase each of these three arms is also a change in how densely the proxies overlap, so they

are compared with arm B at a matched number of neighbours per object, which is recorded in every cell and never at a matched packing fraction. Spherical bodies are not run as a separate arm. A sphere of radius r has the same axis-aligned proxy as a cube of half extent r and a broad-phase operates on the proxy alone, so that at that level arm B already is the sphere case. A body the proxy of which fits loosely is a different matter and arm R provides it. The 52 cells are each run on 30 seeds, 1,560 trajectories in all.

4.3 Protocol

The measurements of the submitted version could not be paired after the fact: their scenes were seeded from the wall clock without the seed being recorded, a fresh scene was drawn for each engine and the integration step was the measured duration of the previous frame, so that a trajectory depended on the engine being timed. Every measurement reported here is therefore new. The integration step is fixed at 1/60 s and a trajectory is determined completely by its arm, its object count, its packing fraction and its seed. Random numbers come from the 32-bit Mersenne Twister [38], seeded from those four values and uniform deviates are formed directly from its output as $u \cdot 2^{-32}$ in double precision, because the generator is specified bit for bit by the language standard whereas the real-valued distributions of the library are not. Arm A accordingly reproduces the distributions of the submitted generator and not its random stream. A trajectory of arm A runs for 1,200 frames, the 20 simulated seconds of the submitted runs and a trajectory of every other arm for 660. The first 60 frames are discarded in every arm, which leaves 1,140 and 600 analysis frames.

Each trajectory is executed seven times from the same seed. One counting pass records the counts of Sub-section 4.4 and compares the complete pair set of every engine with that of BF. The six timing passes, three staged and three outer in alternation, record times from the same source with the counters compiled out, save for the one that the submitted code itself incremented on its timed path, the count of tests that returned true, which every engine retains. A staged pass reads the clock at every stage boundary of an engine, whereas an outer pass reads it only before and after the whole of the work of an engine in a frame. The value of a timed quantity in a frame is the median of its three passes. Total time is always taken from the outer passes and stage times from the staged passes, so that the quantity on which the comparison of net time rests is never perturbed by the stage timers. In every frame of every pass all 14 engines process the same array of bodies, which pairs them by frame and by seed. Their order within a frame is a fresh pseudo-random permutation, drawn from a separate generator seeded by the seed and the pass, so that no engine has a fixed predecessor. After each frame of a timing pass, outside every timed span, a hash of the pair set of each engine is compared with that of BF.

All runs were made on one machine, with an AMD Ryzen 5 7500F processor and 32 GB of DDR5 memory under Windows 11 Pro. The harness has no window and no renderer. It was compiled for x64 as C++17 by Microsoft Visual C 19.51.36246 with the options /O2 /Ob2 /DNDEBUG. Each run of the harness executed one trajectory in a process of its own on a single thread and the timing passes ran at high priority and pinned to one core, the runs following strictly one after another. No build, compilation, analysis or other benchmark ran on the machine during the timing passes, but, contrary to the registered protocol, it was not idle. During 37 minutes of them files were edited, five commits were made and pushed, six bibliographic records were retrieved over the network, text was extracted from four documents and three scripts were checked for syntax; thereafter a status query ran every two minutes until the end, including while the cells timed twice were timed. Descriptively, of the 645 timing runs made before that activity began, 12 carry the flag for instability of Sub-section 4.6 and of the 1,035 made after it began, 70; the cells differ between the two periods, so the contrast cannot be attributed. Time was read from the performance counter of the operating system, the resolution of which on this machine is 100 ns.

4.4 Metrics

The unit of work in a traversal is one evaluation of the box overlap predicate. Evaluations are recorded in three classes: those with at least one internal node as an operand, those between two distinct objects and those of a leaf against itself, which arise only in the symmetric traversal. Their sum is the number of box tests and the number of box tests of the canonical traversal, T, is the primary measure of work. Recorded separately are the evaluations that returned true, the object pairs emitted before deduplication

and the deduplicated candidate pairs E handed downstream, from which the number of neighbours per object is $2E/N$. In every frame the pairs that BF emitted and an engine did not are counted as missing and those that the engine emitted and BF did not as extra. Counts are compared within the hierarchies only: a box test, a comparison in a sweep and a test inside a grid cell are not the same unit, so engines of different families are compared on time alone.

The submitted version reported a quantity that it called pairs checked and described as the number of box tests performed. That description was wrong for the hierarchies. Their counter was incremented after the overlap test had succeeded, so that it recorded the evaluations that returned true and omitted every evaluation that returned false, whereas the counter of the exhaustive engine recorded every evaluation. The two ratios against brute force printed in the submitted version therefore divided tests performed by tests passed and they are withdrawn as reductions in tests whatever the new measurements show. The comparison between the two rules counted both sides in the same way and is not affected in kind. The submitted quantity is retained here under the name passed tests, solely as the bridge to the submitted numbers and the like-for-like replacement for the withdrawn ratios is $N(N-1)/2$ divided by T.

Time is recorded in four stages: the release of the structure of the previous frame, where the form separates it; construction; the traversal, sweep or cell testing that produces the raw pairs; and deduplication, which includes the destruction of the raw list and of the set and is zero by construction for an engine that emits each pair once. In form S, the release of the previous tree falls inside the construction stage, as it did in the submitted code. The total time of a frame is a single span measured in the outer passes. The frame rate reported in the submitted version is not used: the submitted program set no swap interval, truncated the rate to an integer and sampled it once per second and no renderer lies in the measured path of the present harness.

4.5 Statistical Treatment

The unit of observation is one frame and the unit of replication is one seed. The frames of a trajectory are serially dependent and are never treated as replicates; the lag-one autocorrelation and the Durbin-Watson statistic [39] of the per-frame series of T are reported to show the extent of that dependence. The run-level value of a quantity is its mean over the analysis frames of a trajectory and all inference is made on the 30 run-level values of a cell. Because every engine processes the same frames, the two rules are compared within a seed. For a quantity X, the paired value of a seed is the reduction $1 - X_{\mathrm{SAH}}/X_{\mathrm{MED}}$, the ratio $X_{\mathrm{SAH}}/X_{\mathrm{MED}}$ or the relative difference $d = (X_{\mathrm{SAH}} - X_{\mathrm{MED}})/X_{\mathrm{MED}}$, formed from the two rules in the same form. The estimate of a cell is the mean of the paired values over its seeds and never a ratio of cell means.

The interval reported is the 95 percent t interval on the paired values. Where the Shapiro-Wilk test [40] rejects their normality at the 0.05 level in a cell, the percentile interval of a bootstrap [41] over seeds, with 10,000 resamples, governs in its place. The test is correspondingly the one-sample t test of the paired values against their null value, which is 0 for a reduction or a relative difference and 1 for a ratio or the Wilcoxon signed-rank test [42] where normality is rejected. The p values are adjusted by the method of Holm [43] within a family, a family being one question asked of one form in one arm. A cell receives a directional label only if two conditions both hold: the governing interval excludes the null value and the adjusted governing p value is below 0.05. Every other cell is labelled unresolved. No cell is ever labelled equivalent, since a failure to detect a difference is not evidence that there is none. A cell in which the timed quantity under comparison lies, for either rule, at the resolution floor defined in Sub-section 4.6 is unresolved by rule.

The criteria applied to these labels were fixed before the data were collected. The reduction in passed tests reported in the submitted version is re-examined in arm A in direction and in magnitude, the latter by two readings: whether the submitted value lies between the 2.5th and 97.5th percentiles of the 30 paired values and whether it lies between the same percentiles of the 870 values obtained by pairing the heuristic on one seed with the median split on a different seed. The second reading reproduces the unpaired design from which the submitted value was drawn and is the like-for-like one. For construction time, the ratio is said to narrow at a packing fraction if it rises between at most one pair of adjacent object counts, over at least five cells clear of the resolution floor. For net time, the changes of sign of the cell mean of d between adjacent object counts are located by linear interpolation in $\log N$ and all

are reported, but a crossover is claimed only where there is exactly one change of sign and the two cells on either side of it are resolved with opposite labels. Where there is no change of sign, the statement made is confined to the resolved cells of the range measured and nothing is extrapolated beyond it. An arm is compared with arm B at each object count by interpolating the cell means of arm B, linearly in the logarithm of the neighbours per object, to the value observed in the arm. The difference carries a bootstrap interval in which the seeds of the arm and of each cell of arm B are resampled independently and an object count at which the arm lies outside the range of arm B is not evaluable; within a resample that falls outside that range, the interpolation holds the end value. The claim of the submitted version that a uniform scene is conservative for the heuristic is retained only if the reduction in T in the clustered arm exceeds the interpolated value of arm B, with an interval above zero, at five or more of the six object counts. Arms H and R are exploratory and carry no criterion.

4.6 Pre-registration and Instrument Checks

The design, the count model of Section 3, every criterion of Sub-section 4.5 and the consequence attached to each outcome were committed to the project repository before any confirmatory measurement existed and the analysis script was frozen, with its hash recorded, before the first confirmatory row was produced. The instrument was checked on a reserved seed, from which no result is reported and every check that carried a requirement was met without a failure before any confirmatory seed was run: the copies of the submitted engines returned the pairs, their order and the submitted counter of the untouched classes over 4,800 frames of arm A, the counting output was identical byte for byte between two processes in all 52 cells and no pair was missing or extra, nor any structural identity broken, in 36,480 frames. The table of all checks is in the archive of Sub-section 4.10.

Two changes were made to the registered design after the first run of those checks and before any confirmatory seed was run and both were registered in a separate dated record. The packing fraction inside a cluster of arm C was lowered from 1/2 to 1/4, because at the higher value the clustered arm lay above the densest cell of arm B at every object count examined, where the matched comparison could not have been evaluated whatever the data showed. The two cut-off engines of Sub-section 4.1 were added, so that a loss of the heuristic on net time could be attributed either to the rule or to the search that it runs on nodes too small to benefit. They were added after a trial run of the analysis script on reserved seeds had already shown the median split the faster in net time and they carry a secondary hypothesis that cannot alter the primary one. Every check was then repeated from the first on the amended executable and the outcomes stated here are those of that run.

The clock check defines the resolution floor. A timed quantity of an engine in a cell the median value per frame of which on the reserved seed was at or below 20 ticks of the performance counter is treated as unresolvable in that cell and the list of such quantities was fixed before the confirmatory data existed. The total time lies at the floor only for BF and BF-F, in arm A at $N = 50$ and arm B at $N = 50$ and $\phi = 1/32$ and neither the construction time nor the total time of any hierarchy lies at the floor in any cell. The traversal time of the hierarchies in form F does lie at the floor, in three cells of arm B at $N = 50$. Stability between passes is assessed on the confirmatory timing data before any comparison is computed. A trajectory is flagged if the coefficient of variation of its run-level total time over the three outer passes exceeds 5 percent for any engine and where more than 10 percent of the trajectories of a cell are flagged the timing passes of the whole cell are run again once and replace the original, the decision being taken by the script on that statistic alone.

Some quantities reported in this section and in Sections 5 to 7 were not registered. They were computed after the data by a separate script, are counts, differences or ratios of cell means and are called descriptive where they appear: the differences in microseconds, the ratios of construction time between forms and of total time against the exhaustive engine, the passes over the objects made in form F, the shift between the two runs of the cells timed twice, the counts of flagged runs before and after the activity of Sub-section 4.3, the cut-off engines under the narrowing rule and the accounting of stage times, which omits every cell with a stage at the floor; and, for the second campaign, its stage effects, the shifts between the campaigns and the busy time of the machine; and, for the third, the gap between the two inexpensive rules by stage, the cost of a box test and the fronts. The mean time of each stage of every engine in every cell is in the archive.

4.7 A Second Registered Campaign

A second campaign was registered after the data of the first had been read and before either of its two new engines had been run, to test a leaner heuristic. Its rule, SAH1-F, takes the bounds of a node and of its centres in one pass, bins only the axis of greatest centre extent and otherwise keeps the cost, constants, leaf rule, fallbacks and traversal of SAH-F, so that it makes about three passes over the objects at each level of the tree against about eight. SAH1-8-F applies the median split at nodes of 8 objects or fewer, the cut-off of the first campaign, which had itself been added after a trial run. Neither is the builder of Wald [3], the leaves of which hold two to four primitives and the bounds of which are carried through the partition. No engine of the first three campaigns does so; the Morton build of the fourth (Sub-section 4.9) takes the bounds of a node from its children. The campaign ran the 30 cells of arm B on the same seeds, so that every scene is that of the first campaign, timing MED-F, SAH-F and SAH8-F beside the two new engines, five to a frame, with BF-F as the reference of the counting pass. Arm B is where the submitted heuristic did least badly in the first campaign and so the arm most favourable to the new rule. The checks and statistics of the first campaign were applied by an analysis script of its own, frozen before the first confirmatory row, which imports the statistical functions of the first. Which engine governs, the possible outcomes and what this paper would say in each were fixed before the data. No command was issued on the machine during the run, which an author used lightly for browsing; descriptively, the busy time of the whole machine had a median of 1.50 logical processors, below the 2.12 recorded during its checks. The confirmatory run was then repeated two days later, after a restart of the machine, with the executable, the engines of a frame, the cells, the seeds and the frozen script unchanged. Its criterion, whether the 12 cells in which SAH1-8-F had led carry the label again, the wording of each outcome and the rule that it be run once were registered beforehand. The browser and background applications were closed, the run started after ten minutes without keyboard or mouse input and none was recorded during its timing passes; the machine was otherwise idle, save that the update service of the operating system fetched one update and its busy time had a median of 1.25 logical processors.

4.8 A Third Registered Campaign

A third campaign, registered after the data of the first two had been read and before its engine had been run on any seed, set a second inexpensive rule beside the object median. MID-F computes boxes and centres once per frame, as MED-F does and takes the bounds of the boxes of a node in one pass. A node of one object is a leaf. The axis is the longest axis of those bounds, ties going to the lower axis, the plane lies at the midpoint of the bounds on that axis and one pass sends an object the box centre of which lies below the plane to the left child and every other object to the right. If either side is empty, the node is divided at the object median on the same axis and is counted as a fallback. The node array, the leaf rule and the traversal are those of MED-F, so that the two engines differ only in the point at which a node is divided. This split at the midpoint of the node's box on its longest axis, the midpoint split for short, is one form of what is called a spatial median. It is the division that van den Bergen [13] describes in the box divided, the axis, the split point and the point representing an object, save that there an object centred exactly on the plane goes to the other side and an empty side is resolved by two halves of nearly equal size taken without regard to position. He states in one sentence, without a measurement, that the middle of the box performed best in general.

MED-F finds its median with std::nth_element, which in the standard library of Microsoft Visual C++ 19.51.36246 (MSVC STL, _MSVC_STL_UPDATE 202604) sorts by insertion every range of 32 elements or fewer, so that the lowest five levels of every tree are sorted in full; over the trajectories it made 3.2 to 3.9 comparisons per object at each level of the tree, against one for the midpoint split. The campaign timed six engines in a frame, MED-F, MID-F and the submitted and one-axis heuristics with and without the cut-off of 8, in all 52 cells and on the 30 seeds and so the scenes, of the first campaign. It was run once, in one session on the same machine, under the checks and statistics of the second campaign and after MID-F had returned the trees, worked out by hand, of five small configurations; the outcomes and the wording of each were fixed before the data. Programs were closed before the launch at an author's instruction, a mail client and a messaging client being left running. No keyboard or mouse input was recorded from ten minutes before the run to the end of its timing passes, during which the busy time of the machine had a median of 1.26 logical processors. The machine was otherwise idle, save that six notifications of the mail client and one of another application arrived in those passes; leaving

out the 8 trajectories with a busy time above 2.0 changes no label.

4.9 A Fourth Registered Campaign

A fourth campaign, registered after the earlier data had been read and before its engine was run, added a hierarchy built in Morton order [6], [18], the Morton build for short and LBVH-F in the tables. It is built on the processor, in one thread, with a radix sort of three 10-bit digits and the split search of Karras [6]: box centres, quantized to 10 bits per axis on the bounds of the centres of the frame, are sorted by their 30-bit code, a range is divided at the highest bit in which its end codes differ or at its middle if they are equal and the bounds of a node are the union of its children's, not taken in a pass over its objects as in every other hierarchy here. Its node array, leaf rule and traversal are those of MED-F. It was timed with MED-F, MID-F and SAH1-8-F in all 52 cells on the 30 seeds of the first campaign, once, in one session on the same machine, after it had returned five hand-worked trees and matched a separately written builder on 2,000 random configurations. In the timing passes, run in the evening from 18:43 to 19:33 with the other computations of the machine stopped, no keyboard or mouse input was recorded and the median busy time was 1.40 logical processors, so that by the earlier criterion the machine was otherwise idle; leaving out the 136 of 1,560 trajectories above 2.0 (90th percentile 1.92) changes none of the 208 labels that compare the Morton build or the variant with the object median. The executables of every campaign were compiled by Microsoft Visual C++ 19.51.36246 with its own standard library.

4.10 Data and Code Availability

The harness, the list of seeds, the run-level tables, the analysis script, the registered design and the tables that the text describes and does not print are available at https://github.com/RazordousJR/BVH_CollisionDetection and in the archive supplied with this submission. That archive, with the registration and records of the repeat session and of the third and fourth campaigns, is also deposited beside the per-frame rows of every trajectory, which are too large for the submission system, at one anonymous address that needs no account: https://osf.io/8m6pk/overview?view_only= 0c85355d515a4d45aebb20418291dcfb. Every value in the tables and figures of this paper is computed from those files by script, the script of the descriptive analysis being in the archive and none is transcribed.

5. RESULTS

Every one of the 1,560 trajectories completed its counting pass and its timing passes. In all 1,094,400 frames no engine missed a pair that BF emitted and none emitted a pair that BF did not, the pair stream of every timing pass was that of the counting pass and the structural identities held in every frame to which they apply (Table 3, which also counts the frames with a tie of volumes, of which Equation (4) is not required). The check of stability between passes sent four of the 52 cells to a second run and three of them, arm B at $N = 50$ and $\phi = 1/2$, arm B at $N = 200$ and $\phi = 1/2$ and arm C at $N = 1{,}000$, remain over the bar after it and are reported with that flag. The total of a staged pass differed from that of an outer pass by a median of between -0.13 and 0.31 percent according to the engine and by at most 2.2 percent for one engine in one cell. All 564 comparisons of cells reported below carry a directional label; the bootstrap interval and the signed-rank test govern in 178 of them. The intervals are intervals over seeds within one session of measurement. As a descriptive check, between the first and the second run of the cells that were timed twice, on the same seeds, the mean of a comparison moved by up to 3.8 percent of its value in forms S and C and by up to 1.1 percent in form F and the second mean lay outside the first interval in nine of 32 comparisons, none of which changed sign.

The heuristic performs fewer box tests than the median split in every cell. The reduction in T lies between 17.8 and 46.0 percent over arm B and between 25.8 and 46.7 percent over arm A. Table 4 separates the classes of test in arm A. The candidate pairs are the same under both rules, as Sub-section 3.1 requires and the exhaustive engine performs between 7.0 and 19.4 times the box tests of the median split and between 13.2 and 26.2 times those of the heuristic. Fewer tests are not less time, however, as a descriptive comparison of cell means shows. At N of 100, 200 and 500, where the time of the exhaustive engine is clear of the resolution floor, the heuristic in form F takes 4.78, 1.59 and 0.75 times the total time of BF-F and the median split 1.42, 0.59 and 0.32 times and in the as-submitted form, the heuristic takes 5.71, 1.93 and 1.00 times the total time of BF and the median split 2.89, 1.23 and 0.77

times. At every object count, the reduction falls as the packing fraction rises. Against the object count, it falls at the three lower packing fractions, where Spearman's coefficient lies between -1.00 and -0.83 and not at the two higher, where it is 0.09 and 0.77. The submitted version reported an advantage that declined with the object count; in its generator, the density rose with the object count and the two effects could not be told apart. In arm A, the reduction in passed tests that the submitted version reported is replicated in direction at all four object counts (Table 5). Its magnitude is replicated in part: the submitted value lies within the paired range at $N = 50$ and 100 and outside it at 200 and 500, at $N = 500$ by the third decimal alone (17.7 percent against a range that begins at 17.72) and within the unpaired range, which reproduces the design of the submission, at all four.

Table 3. Agreement with the exhaustive reference, determinism and structural identities over all 1,094,400 frames of the confirmatory trajectories.

Check	Count	Check	Count
Pairs missing from, or extra to, the set of BF, each of the 13 other engines	0; 0	Form S: self tests not N or raw pairs not $2E$	0
Trajectories, counting and timing passes	1,560; 1,560	Equation 4 fails, no volume tie	0
Pair stream differs between counting and timing	0	Frames with a volume tie	612,712
Timing passes disagree with one another	0	Equation 4 fails among those: MED; SAH	58; 50
Pair-set hash differs from BF on the timed path	0	T differs between forms C and F, no median tie	0
Internal nodes not $N - 1$ or leaves not N	0	Frames with a median tie	153
Median split: passes not $N + 2D$ or $D \neq s(N)$	0	T differs among those: MED; SAH	26; 0
Heuristic: Equation 3 fails or merges not 22 per axis	0	Third campaign, 1,094,400 frames: pairs missing; pairs extra; failures of the eleven structural checks	0; 0; 0

Counts are trajectories in the second row and frames, summed over the two rules, in the rows below it; the counts of each engine are in the archive and the identities are those of Sections 3.2 and 4.1.

Table 4. Box tests of the canonical traversal by class and candidate pairs, in arm A.

N	Rule	Internal-node tests	Object-pair tests	T	Candidate pairs E	All pairs	All pairs over T
50	Median	141.0	34.3	175.3	5.1	1,225	7.00
	Heuristic	68.6	24.9	93.5	5.1	1,225	13.16
100	Median	432.8	94.4	527.2	21.7	4,950	9.40
	Heuristic	239.8	65.1	304.9	21.7	4,950	16.30
200	Median	1,278.9	271.7	1,550.6	84.9	19,900	12.85
	Heuristic	809.8	190.8	1,000.6	84.9	19,900	19.92
500	Median	5,268.3	1,151.0	6,419.2	528.4	124,750	19.44
	Heuristic	3,815.3	950.6	4,765.9	528.4	124,750	26.19

Means per frame in form C. An internal-node test has at least one internal node as an operand and an object-pair test has two distinct objects; T is their sum. All pairs is $N(N-1)/2$, the number of tests of the exhaustive engine and the last column, the like-for-like replacement for the two withdrawn ratios, is the mean over seeds of the per-seed ratio and not the ratio of the column means.

Table 5. Re-examination in arm A of the reduction in passed tests reported in the submitted version, for the as-submitted engines.

N	MED-S	SAH-S	Reduction, %	p	Submitted, %	Paired range, %	Unpaired range, %
50	333.5	251.7	24.5 [24.2, 24.8], t	<0.001	24.0	[22.9, 26.2], yes	[17.5, 30.3], yes
100	868.7	646.3	25.6 [25.3, 25.9], t	<0.001	26.7	[24.2, 26.8], yes	[18.1, 32.1], yes
200	2,318.5	1,768.4	23.7 [23.5, 23.9], t	<0.001	19.6	[22.7, 24.6], no	[17.7, 29.3], yes
500	8,974.1	7,320.8	18.4 [18.3, 18.5], t	<0.001	17.7	[17.72, 18.92], no	[14.4, 22.0], yes

MED-S and SAH-S are mean passed tests per frame. Every row carries the label heuristic fewer. The reduction is given with its governing 95 percent interval and the method that governs. The paired range

is the 2.5th to 97.5th percentile of the per-seed reductions and the unpaired range the same percentiles over all ordered pairs of distinct seeds; each is followed by whether it contains the submitted value.

Construction by the heuristic is slower in every cell of every form (Figure 2). Over arm B, it takes between 3.25 and 5.91 times as long as construction by the median split in form F and between 1.32 and 3.04 and between 1.25 and 2.79 times in forms C and S, where both rules also pay for the allocation of nodes and the copying of index lists. At every packing fraction and in every form, the ratio falls from each object count to the next, which is narrowing in the sense fixed in advance and, as Sub-section 3.2 warned, cannot by itself confirm the model. The counts behave as modelled. The heuristic skipped an axis at fewer than two in a million of its opportunities in every cell, so that its fixed work was 99 operations per internal node to four significant figures and it fell back to a median division at between 3.2 and 29.8 percent of the internal nodes; the comparisons made by the median split are tabulated against their reference in the archive. Under an equal weighting of every operation, node to four significant figures and it fell back to a median division at between 3.2 and 29.8 percent of the internal nodes; the comparisons made by the median split are tabulated against their reference in the archive. Under an equal weighting of every operation, the count ratio is 3.52 to 5.18 in form F and 1.75 to 4.24 in forms S and C in arm B; no agreement with the ratios of time was predicted. Read descriptively against a related engine, the narrowing need not be a convergence of the two rules: with the cut-off of 8 objects the ratio, 1.66 to 2.02, rises between at least one pair of adjacent object counts at every packing fraction.

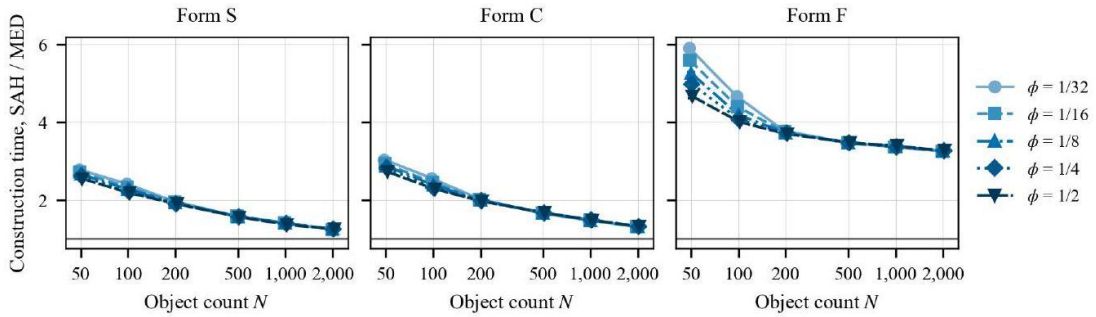


Figure 2. Ratio of the construction time of the heuristic to that of the median split against object count, for each packing fraction of arm B and in each form. Markers are cell means of the per-seed ratio, bars are the governing 95 percent intervals and the horizontal line marks a ratio of 1.

The heuristic as submitted does not repay that construction anywhere in the range measured. The median split is the faster in net time in every cell of arm B and in every form and the cell mean of d changes sign nowhere, so that no crossover lies between $N = 50$ and $N = 2{,}000$ (Figure 3 and Table 6). The size of the difference depends on the form: d lies between 0.09 and 1.41 in form S, between 0.21 and 1.65 in form C and between 1.11 and 3.65 in form F.

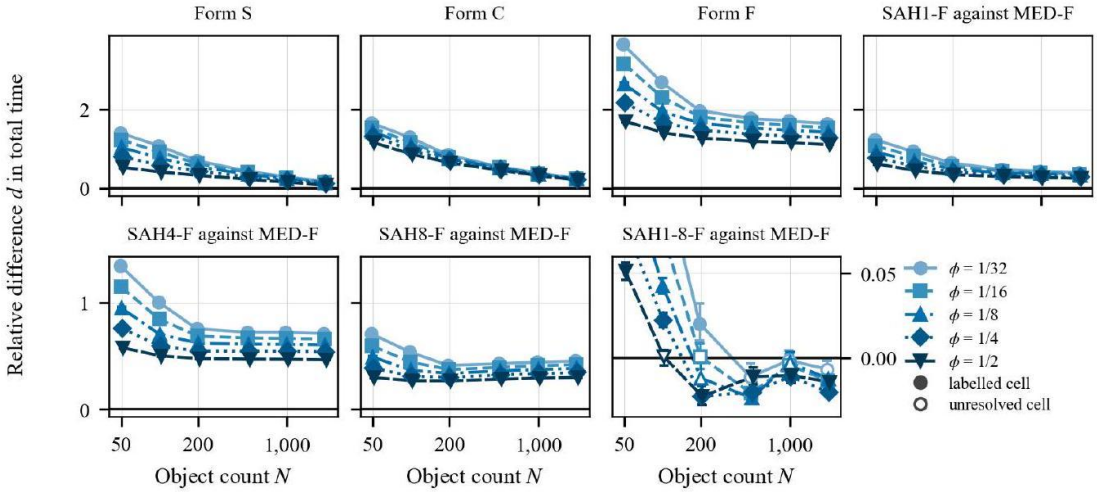


Figure 3. Relative difference d in total time against object count in arm B: the forms and cut-off engines of the submitted rule and the one-axis engines of the second campaign (the last panel on its

own scale, cut at 0.06 above the cells of 50 and some of 100 objects). Below the line $d = 0$ the heuristic is the faster; hollow markers are unresolved cells and bars the governing 95 percent intervals.

Two descriptive quantities set those ranges in context. In absolute terms, the heuristic costs 16 to 18 microseconds more per frame than the median split at $N = 50$ in all three forms and at $N = 2{,}000$ it costs 222 to 264 microseconds more in form S, 319 to 329 in form C and 687 to 708 in form F. And form F did not treat the rules alike: against form C, it builds the median split 2.27 to 3.43 times faster and the heuristic 1.32 to 1.39 times faster. In every form, d falls as the object count rises and as the packing fraction rises. In form S, it is between 0.09 and 0.18 at $N = 2{,}000$ and still falling. A cross-over above 2,000 objects is therefore neither located nor excluded in any form and nothing is extrapolated. The label is the same in all 30 cells for every pair of forms. d is lower in form S than in form C by 0.07 to 0.62 in every cell, in the direction Sub-section 4.1 anticipated; form S differs from form C in its traversal, in its deduplication and in where it releases the previous tree and these effects are not separated. It is lower in form C than in form F by 0.54 to 1.99. The two cut-off engines are slower than the median split in every cell as well, with d between 0.47 and 1.35 for a cut-off of 4 objects and between 0.27 and 0.71 for a cut-off of 8. Withholding the search from nodes of up to 8 objects therefore leaves the direction of the result unchanged and removes most of its size in form F; no larger cut-off was tested. In arm A, which reproduces the submitted generator, the median split is likewise the faster in every cell, with d between 0.30 and 1.43 in form S, 0.49 and 1.66 in form C and 1.35 and 3.75 in form F.

Table 6. Labels of the comparison of total time over the cells of arm B, by form and the outcome of the cross-over rule at each packing fraction.

Form	Heuristic faster	Median faster	Unresolved	1/32	1/16	1/8	1/4	1/2
S (SAH-S against MED-S)	0	30	0	M	M	M	M	M
C (SAH-C against MED-C)	0	30	0	M	M	M	M	M
F (SAH-F against MED-F)	0	30	0	M	M	M	M	M
F with cut-off (SAH4-F against MED-F)	0	30	0	M	M	M	M	M
F with cut-off (SAH8-F against MED-F)	0	30	0	M	M	M	M	M

The last five columns are the packing fractions. M: the median split is the faster in every resolved cell. H: the heuristic is the faster in every resolved cell. X: a cross-over is claimed, at the interpolated object count shown. S: the sign of d changes without the conditions for a cross-over being met. U: no cell is resolved.

In the second campaign, no pair was missing or extra in any of 594,000 frames and the three shared engines reproduced every count of the first. The one-axis heuristic built in 0.44 to 0.50 times the time of the submitted rule, needed 3.3 to 8.7 percent more box tests and took 41 to 54 percent less net time; against the median split, it remained the slower in all 30 cells, with d between 25 and 124 percent (Figure 3). With the median split at nodes of 8 objects or fewer, the one-axis heuristic was the faster in 12 cells, by 1.1 to 2.4 percent of net time (500 and 2,000 objects at packing fractions 1/16 to 1/2; 200 and 1,000 objects at 1/4 and 1/2), the slower in 10 and unresolved in 8. A repeat session, reported below, reproduced the label in 11 of the 12. In those cells, descriptively, its construction was 5 to 10 percent slower than the median split's and its traversal 10 to 25 percent faster, a saving of 0.9 to 11 microseconds per frame. The cell mean of d changes sign once at each packing fraction, from 1/32 to 1/2 near 360, 206, 172, 140 and 102 objects in this session and near 344, 283, 186, 151 and 98 in the repeat. The rule fixed in advance locates a cross-over between 100 and 200 objects at 1/4 in both and between 200 and 500 at 1/32 in the repeat alone, so no cross-over is stated beyond the session that located it; d does not fall steadily with the object count. Between the first campaign and this one, which differ in the engines of a frame as well as the session, the shared engines kept every label and the net time of the median split differed, descriptively, by up to 8.7 percent in those cells, more than the margin. In the repeat no pair was missing or extra and the counting output was identical byte for byte with that of this session. SAH1-8-F was again the faster in 11 of the 12 cells, by 1.1 to 2.4 percent there in both sessions; the twelfth, 1,000 objects at 1/4, was unresolved, with d of -0.6 percent (interval -1.1 to 0.0), so that whether the variant repays its construction there is not established and the advantage is claimed only for the cells labelled in both sessions. Corrected over those 12 cells alone, a weaker test, 11 carry the label. Between

the sessions, the cell mean of d moved by -0.3 to 0.8 percentage points in those cells and the net time of the median split by at most 0.4 percent. The smallest margin is larger than either movement, yet a movement of 0.6 points, half the margin of its cell, lost the label of the twelfth and three of the 30 labels of this engine differ between the sessions, SAH1-8-F being the faster in the repeat alone at 500 and 1,000 objects at 1/32; the other engines kept every label. Descriptively, in these two sessions, the difference takes one of two values in a process: in most processes of a cell the variant is 1.4 to 2.8 percent faster and in the rest 0.4 to 2.5 percent slower, its construction 4 to 9 percent longer, the state lasting the process. A cell mean therefore reflects the share of each that a session drew, 51 and 67 of 360 processes being in the slower state in the two sessions.

In the third campaign, no pair was missing or extra in any of 1,094,400 frames. A split at the midpoint of the node's box was faster than the object median in 30 of the 30 density-controlled configurations, by 21.0 to 39.3 percent of net time and in every cell of the replication arm, the clustered arm, the arm of heterogeneous sizes and the arm of loose proxies (Table 7); these are sizes of the object median as implemented with the library's selection. The stage times show how the difference is made up. The midpoint tree needed 9.5 to 32.0 percent fewer box tests and, in the 48 cells clear of the resolution floor, was built in 0.54 to 0.63 times the time. Descriptively, from the cell means of the stage times, construction is 76 to 90 percent of the gap in every cell. With construction set equal the midpoint split would still lead, by 4.0 to 8.0 percent in arm B and by 3.5 to 12.5 percent over all arms, on its traversal alone, which took 6 to 24 percent less time in 52 of the 52 cells, five of them at the resolution floor. That is not its saving in box tests: a box test cost it 1 to 21 percent more. MID-F fell back to the object median at 0.31 to 1.37 percent of its internal nodes, by arm.

Table 7. The third campaign by scene arm: the midpoint split (MID-F) against the object median (MED-F) and the one-axis heuristic with the cut-off (SAH1-8-F) against the midpoint split.

Arm	Cells	d of MID-F against MED-F, %	Fewer box tests, %	Construction time, MID-F over MED-F	d with construction set equal, %	d of SAH1-8-F against MID-F, %	Fallback of MID-F, %
A	4	-35.3 to -29.7	12.7 to 32.0	0.57 to 0.58 (1)	-8.0 to -5.7	39 to 82	0.32
B	30	-39.3 to -21.0	9.5 to 31.6	0.55 to 0.63 (2)	-8.0 to -4.0	32 to 80	0.31
C	6	-38.0 to -23.7	15.5 to 24.3	0.54 to 0.63 (0)	-12.5 to -3.5	46 to 52	0.57
H	6	-36.2 to -29.8	15.9 to 22.7	0.56 to 0.58 (1)	-8.0 to -5.2	48 to 65	0.97
R	6	-28.2 to -22.5	9.8 to 13.4	0.56 to 0.62 (0)	-7.2 to -4.0	34 to 40	1.37

Ranges are of cell means over the cells of an arm and the arms are those of Table 2. d is the relative difference in total time and is negative where the engine named first is the faster. The construction ratio is over the cells clear of the resolution floor, the number of cells at the floor being in parentheses. d with construction set equal is descriptive: from the cell means of the stage times, the traversal time of MID-F less that of MED-F, over the construction time of MID-F plus the traversal time of MED-F. The fallback is the share of the internal nodes of MID-F divided at the object median, pooled over the arm.

The question of this paper can therefore be put to the faster of the two inexpensive rules as implemented. No heuristic tested, as built, was faster than the midpoint split in any density-controlled configuration or in any cell of another arm and no cell was unresolved. In arm B the one-axis heuristic with the cut-off, the nearest of the four, took 32 to 80 percent more net time than the midpoint split, the submitted heuristic with the cut-off 70 to 166, the one-axis heuristic 71 to 232 and the submitted heuristic 190 to 629. The 12 configurations in which the variant with the cut-off led the object median in the second campaign are among them.

Against the object median, the labels of the four heuristics can be set beside those of the earlier campaigns. In the third campaign, the frames of which hold six engines, the variant was labelled faster than the object median in 12 of those 12 configurations, slower in 0 and unresolved in 0. In arm B it is

so labelled in 20 cells, slower in 9 and unresolved in 1; 9 of those labels differ from the second campaign's and 8 from its repeat's, under a change of frame and of session that cannot be told apart and the other heuristics kept every label. Each of the 20 has a margin below 4 percent, as have 34 of its 51 labels over all arms (0.4 to 3.8 percent) and a margin of 1.1 percent lost its label in the repeat of the second campaign. Elsewhere, where the labels against the object median are a by-product of the frame, the variant was labelled faster in 2 of the 4 cells of arm A and 4 of the 6 of each of arms C, H and R and each of the other three heuristics slower in every cell.

By the registered division of the processes of a cell into two states, 0 of the 52 cells count as two-valued for either new comparison and none for the variant against the object median. The two states reported above are therefore confined to the second campaign and its repeat: in the third, the state was absent from that engine and present in another (SAH8-F, in 18 cells of arm B by the first two conditions of the rule), so that it follows the executable and not the rule; its cause was not identified. Figure 4 sets mean construction time against mean traversal time. An engine is dominated in a cell if another has both means lower, a description of means in which cells at the resolution floor are not counted. Of the 3, 26, 6, 5 and 6 cells left in arms A, B, C, H and R, MID-F is on the front in all and has the least sum of the two means in all, MED-F is on the front in none and SAH1-8-F is on it in 3, 26, 6, 5 and 6, SAH1-F in 3, 24, 5, 4 and 5, SAH8-F in 3, 15, 4, 3 and 4 and SAH-F in 2, 19, 4, 3 and 4, none of them with the least sum in any, a count on the front not weighing the two times against one another.

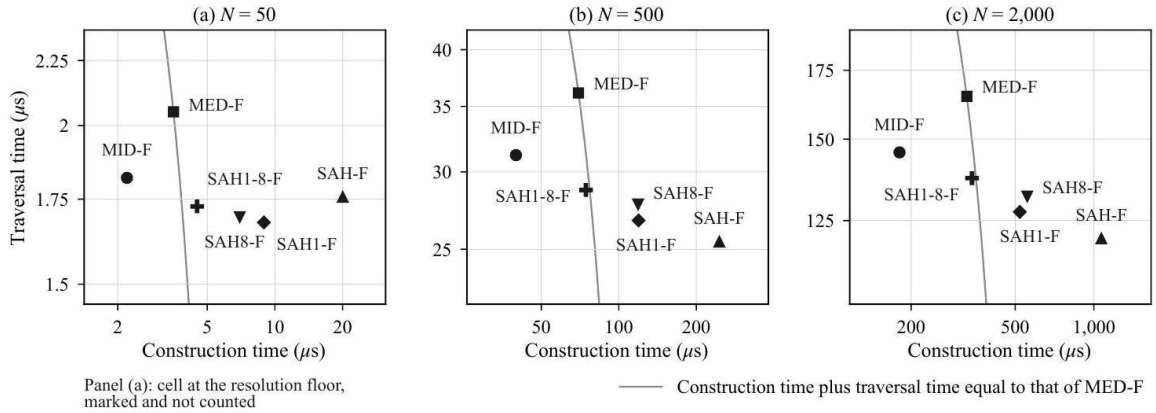


Figure 4. Mean traversal time against mean construction time of the six engines of the third campaign in arm B at a packing fraction of 1/8. The curve joins the points the two means of which sum to those of MED-F; the cell of the first panel lies at the resolution floor.

In the fourth campaign, no pair was missing or extra in any of 1,094,400 frames and the carried engines counted as in the third in every frame. Table 8 gives times at three object counts; the ranges by arm are in the archive. Against the midpoint split, the Morton build was faster in net time in 26 of the 30 density-controlled configurations, every one of 100 objects or more and that of 50 objects at 1/2, by 0.6 to 36.9 percent and slower in the other 4, of 50 objects at 1/32 to 1/4, by 2.9 to 18.1 percent. In each other arm, it was faster at 100 objects and above and slower at 50, the midpoint split leading in 8 of the 52 cells, all of 50 objects. Every label is the result of one session on one machine; four margins are below 4 percent, 2.9 and 0.6 percent at 50 objects and 1/4 and 1/2, 1.0 at 100 objects in the clustered arm and 3.1 at 50 objects in the arm of loose proxies and a margin of 1.1 percent lost its label in the repeat of the second campaign.

Against the object median, the Morton build was faster in net time in all 30 density-controlled configurations, by 20.2 to 60.7 percent and in every cell of the other arms. In no density-controlled configuration was the one-axis heuristic with a cut-off labelled faster than the Morton build: it was slower in all 30, by 38 to 152 percent and in every other cell; this concerns that heuristic alone, the other three not being in the frame. The Morton tree needed 3.9 to 8.5 percent more box tests than the midpoint tree in the density-controlled arm and 9.9 to 15.1 percent in the clustered arm. It was built in 0.43 to 1.09 times the time of the midpoint split in the 27 density-controlled configurations clear of the floor (3 at it), longer only in 2 of 50 objects and in 0.24 to 0.80 times that of the object median. The construction times of the Morton build include a fixed cost of clearing and scanning 3,072 buckets of its radix sort in every frame, which weighs most at the smallest object counts and its node bounds are taken from its

children rather than in a pass over the objects of the node; a difference in construction between it and the midpoint split is therefore a difference between these two implementations and not of Morton order alone. Descriptively, at 500 objects and above, its traversal took longer in 14 of 15 such configurations, by up to 10.7 percent, so its whole advantage there lies in construction, which made up 56 to 96 percent of it at 100 and 200 objects; with construction set equal, d would lie between -3.2 and 6.3 percent. Half of the passes of the midpoint split over the objects take the bounds of a node, which the Morton build does not do and the share of Morton order itself was not measured.

Table 8. The fourth campaign in arm B at a packing fraction of 1/8: mean construction time and mean net time per frame, in microseconds, of the object median (MED-F), the midpoint split (MID-F) and the Morton build (LBVH-F) and the relative difference d in net time of the Morton build against each. These means are of the fourth campaign and differ from those of the third campaign for the same cells (Figure 4).

N	MED-F Constr.	Net	MID-F Constr.	Net	LBVH-F Constr.	Net	d of LBVH-F, %	
							Against MID-F	Against MED-F
50	3.44	5.36	2.16*	3.91	2.51 [2.50, 2.51]	4.18 [4.14, 4.23]	7.0 [6.4, 7.6] midpoint faster	-22.0 [-22.4, -21.6] Morton faster
500	67.10	101.48	38.59	68.11	19.54 [19.51, 19.58]	50.39 [50.17, 50.61]	-26.0 [-26.2, -25.9] Morton faster	-50.3 [-50.5, -50.2] Morton faster
2,000	322.84	483.02	179.56	320.32	77.28 [77.14, 77.43]	230.92 [230.39, 231.44]	-27.9 [-28.0, -27.8] Morton faster	-52.2 [-52.3, -52.1] Morton faster

*At the resolution floor of the clock, the mean is printed and a comparison resting on it is unresolved.

Means over the 30 seeds of the run-level values: Beneath each mean of LBVH-F and beneath each d, is its governing 95 percent interval (t or the percentile bootstrap where the Shapiro-Wilk test rejects normality) and each d carries its label. d is negative where the Morton build is the faster.

Beside the third campaign, the labels against the midpoint split were unchanged. Against the object median, the variant led in 10 of the 12 configurations where it had led in the second campaign and was slower in 2, both of 200 objects and among the 11 for which its advantage is claimed; its margins at 500 objects were 0.36 to 0.71 percent, against 1.9 to 2.5 percent in the third campaign and 12 of its 52 labels differ, a change of frame and of session not told apart. With the counted work identical, the traversal of the object median was 3.3 to 8.4 percent shorter than in the third campaign and that of the variant at most 3.1 percent shorter, so the margin follows the session or the frame, not the rule, for no identified cause; no cell of arm B counts as two-valued.

In the clustered arm, the reduction in T exceeds the value interpolated from arm B at the same number of neighbours per object, with an interval above zero at five of the six object counts, by 3 to 10 percentage points, the exception being $N = 50$ (the table is in the archive). By the criterion fixed in advance, the claim that a uniform scene understates the benefit of the heuristic is supported and it is supported in that sense alone. Against arm B at the global packing fraction of the clustered arm, 1/32, the reduction in the clustered arm is the smaller at every object count, by 3 to 23 percentage points. On net time, for which no criterion was fixed, the matched comparison goes the other way: in form F, d in the clustered arm exceeds the interpolated value of arm B at every object count, by 0.18 to 0.26, so that the clustered scenes are the less favourable to the heuristic in time. In arms H and R, the heuristic again performs fewer box tests and is again the slower in net time in every cell, with d between 1.50 and 3.08 and between 1.15 and 1.96 in form F. At matched neighbours per object, the reduction in T of arm R cannot be told from that of arm B at any object count and that of arm H is lower at the three smallest object counts, higher at $N = 1{,}000$ and indistinguishable at the others.

Among the 14 engines of the first campaign, no hierarchy has the lowest mean total time in any of the 52 cells (Table 9). The exhaustive engine on boxes computed once per frame is the fastest at $N = 50$ and a uniform grid or the sorting baseline at every larger object count, the grid named being the best of three cell sizes chosen after the fact; the engine named leads in every seed in 40 of the 52 cells. These

figures position the hierarchies and are not a finding about the families. Both baselines are rebuilt in every frame without the coherence between frames that the methods they stand for exploit.

Table 9. The engine with the lowest mean total time in each cell.

N	A	B, 1/32	B, 1/16	B, 1/8	B, 1/4	B, 1/2	C	H	R
50	BF-F (24)	BF-F (29)	BF-F (30)	BF-F (30)	BF-F (30)	BF-F (30)	BF-F (30)	BF-F (30)	BF-F (30)
100	G2.0 (22)	G2.0 (30)	G2.0 (30)	G1.5 (28)	SAP (27)	SAP (30)	SAP (30)	G1.0 (28)	SAP (30)
200	G1.5 (30)	G2.0 (30)	G1.5 (30)	G1.0 (14)	G1.0 (30)	G1.0 (19)	SAP (28)	G1.0 (30)	G1.0 (30)
500	G1.0 (30)	G2.0 (30)	G1.5 (30)	G1.0 (30)	G1.0 (30)	G1.0 (30)	G1.5 (24)	G1.0 (30)	G1.0 (30)
1,000		G2.0 (30)	G1.5 (30)	G1.0 (30)	G1.0 (30)	G1.0 (30)	G1.5 (17)	G1.0 (30)	G1.0 (30)
2,000		G2.0 (30)	G1.5 (30)	G1.0 (30)	G1.0 (30)	G1.0 (30)	G1.5 (27)	G1.0 (30)	G1.0 (30)

In parentheses, the number of seeds in which that engine was the lowest. Columns are the arms, arm B being divided by packing fraction. G1.0, G1.5 and G2.0 abbreviate the grids GRID-1.0, GRID-1.5 and GRID-2.0.

Within a run the frames are far from being independent. Across the cells, the lag-one autocorrelation of the per-frame series of T has a median over seeds of 0.62 to 0.96 for the median split and of 0.75 to 0.97 for the heuristic and the Durbin-Watson statistic a median of 0.07 to 0.76 and of 0.05 to 0.50, against the value of 2 of an uncorrelated series.

6. DISCUSSION

For the rule that was submitted and in the range measured, the question of the Introduction has a plain answer: rebuilt in every frame over box proxies, it did not repay its construction against the object median in any cell, in any form or with a cut-off of 4 or 8 objects. A descriptive accounting, not registered and formed from the stage means of form F in the 27 cells of arm B in which every stage is clear of the resolution floor, runs as follows. Under the median split, construction already takes between 49 and 76 percent of a frame. The heuristic shortens the traversal by between 17 and 40 percent and the time so saved is between 2.8 and 7.7 percent of the construction time that the heuristic adds. Put as a requirement, this rule would have broken even in this form only if its construction had cost no more than 1.09 to 1.21 times that of the median split, whereas in the same cells it costs 3.25 to 4.98 times and, with the cut-off of 8, 1.66 to 1.81 times against a requirement of 1.08 to 1.15. The requirement belongs to the form and to the tree: it is 1.01 to 1.05 in form C and 1.04 to 1.11 in form S.

The result for the submitted rule is not a verdict on binned surface area builders in general and the second campaign does not settle the general case either. The submitted rule is a costly member of its family, binning three axes where the builder of Wald [3] bins one and searching down to two objects. A one-axis rule built in about half its time and, applied at every node, still lost to the median split in every cell; only with the search also withheld from small nodes did it win, in twelve cells and by at most 2.4 percent, its slower construction outweighed by a faster traversal. The win is narrow and is not a property of every run: a repeat session reproduced it in eleven of the twelve cells and about one process in six showed the variant the slower, by a state of the process the cause of which was not identified. Its margin is larger than the movement between those sessions of the median split's own net time and of the difference itself, but half a margin of movement lost a label; it is not presented as a practical performance advantage. In a fourth session, with four engines in a frame, it led in ten of the twelve, by below 1 percent at 500 objects and was slower in two at 200 objects, for no identified cause. No novelty is claimed for combining a heuristic with a cheaper split at small nodes. The margin is a few microseconds per frame, whereas the fastest baseline of the first campaign took less time than the median split in every cell, by a factor of 1.13 to 2.90, so no hierarchy is recommended here. The Morton build of the fourth campaign was not timed beside the baselines. For the submitted rule, the measurement agrees with the position that the collision literature states [6]-[7].

The third campaign obliges this paper to raise an objection against itself: through two campaigns the heuristics were measured against the slower of two inexpensive rules, the object median not being the faster of the two in this regime as implemented. A heuristic is behind the midpoint split where it is so labelled, which here is every cell and the twelve cells of the variant with the cut-off are cells of an advantage over the object median alone. No order of the two rules as such follows: most of the gap is

the cost of the library's selection and why the midpoint tree is traversed in less time was not examined. The fourth campaign extends the objection: there, the midpoint split was itself slower than the Morton build at 100 objects or more. Set now beside three inexpensive rules, the heuristics led only the object median and the variant with a cut-off, the only heuristic timed beside the Morton build, trailed it in every cell.

The recommendation of the submitted version rested on a metric that counted only successful tests and on form S, which lowers *d* by 0.07 to 0.62 against the control; even in form S, the median split is the faster in every cell.

7. LIMITATIONS

The conclusions hold within the following limits, most of which were declared before the data were collected. Absolute times and with them the position of any crossover, are properties of an implementation, which is why three forms are run. Forms S and C allocate and copy at every node, a cost common to both rules that is expected to draw the ratio of their construction times towards one and form S carries the difference in the release of the previous tree inside its construction stage. Forms S and C share one construction code, so that the three forms are two implementations of construction and the conversion to form F, which changed several things at once, sped the median split up more than the heuristic. In form F, the submitted heuristic still makes about eight passes over the objects at each level of the tree, against about one for the median split; the second campaign tested a three-pass rule on one axis and no three-axis rule in fewer passes. Every tree is traversed once, so nothing is concluded about a hierarchy that is refitted or queried repeatedly between constructions. The bodies pass through one another freely, so the dense cells contain configurations that a simulation with contact would not sustain and the density of arm A is not stationary within a run. All scenes are synthetic and made of box proxies, with no triangle mesh behind them and no public suite, such as Broadmark [2], was run, so the conclusions are confined to the workloads tested. A grid sized to the largest proxy is disadvantaged where sizes vary and is reported uncorrected. The engines of a frame share one array of bodies and one heap and the random order in which they run spreads the influence of one engine on the next without removing it. The intervals are intervals over seeds within one session. The margins of the second campaign are differences within one frame of one session and each is the mean of two states of a process (Section 5). Between the first two campaigns the net time of the median split moved by more than those margins and between the second campaign and its repeat by up to 0.4 percent in the cells at issue; reproducibility is established no further than one repeat on one machine. The one-axis engines were compiled as a separate template from the median split, with a traversal identical in source and a difference of about one percent in the code generated is not excluded; in those two sessions the two states lay in the construction time of these engines, in the third campaign in that of another and their cause was not identified.

The study was made on a single machine, a known limitation of every timing result in it: no second processor was available, so neither the direction of the comparison of net time nor any margin has been tested on other hardware and the repeat session ran on the same machine. One compiler, one standard library and one thread were used. The counts are independent of the hardware by construction and are released with the seeds; the times are those of one processor. Two limits concern the rules themselves. As implemented, the object median was the slower of the two inexpensive rules measured and the size of the gap is not a property of the rule: an object median with a leaner selection was not measured and every comparison with the object median in this paper is with this implementation of it. Of the rules that divide at a point of space, the Morton build apart, only the midpoint of the node's box was measured and not the midpoint of the bounds of the centres or the mean of the centres [4]. A hierarchy built in Morton order [6], [18], which here quantizes the box centres on the bounds of the centres of each frame and divides a node at the middle of a grid cell on an axis set by the position of the bit, was measured in one implementation only: on the processor, in one thread, with a radix sort of three 10-bit digits, leaves of one object and bounds taken from the children. A parallel build, a build on a graphics processor, on which its authors timed it, a comparison sort at small object counts, a grid fixed over the whole domain of a scene and the refinements of later work [19] were not measured. The heuristics fall back to the object median, so that their comparison with the midpoint split describes them as built. The heuristic is the surface area form, with twelve bins on each of three axes and a unit traversal cost, as submitted; the

volume criterion of the collision literature [4], [14], other numbers of bins and cut-offs and leaves of more than one object were not tested. The second campaign and its repeat ran arm B only and timed five engines in a frame instead of fourteen; the object count at which a sign changes depends on those conditions. The third campaign is one session on the one machine, with six engines in a frame. The fourth campaign is one session on the one machine, with four engines in a frame. Finally, only axis-aligned boxes are used. A tighter volume reports fewer false overlaps, at a construction cost that has been measured [44] as the highest among the volumes compared there.

8. CONCLUSIONS

This paper measured, on paired and seeded scenes and under criteria fixed in advance, whether a binned surface area heuristic repays its construction against an object median split when a broad-phase hierarchy over box proxies is rebuilt in every frame. The heuristic as submitted, which searches twelve bins on three axes at every node, does not: not at any object count from 50 to 2,000, at any of the five densities, in any of the three forms of implementation or with its search withheld from small nodes, although its tree needs 17.8 to 46.0 percent fewer box tests. A one-axis heuristic applied at every node does not repay it either. With the search also withheld from nodes of 8 objects or fewer, the one-axis heuristic is faster than the median split in 12 of the 30 cells of arm B, by at most 2.4 percent of net time, slower in 10 and unresolved in 8. A repeat session on the same machine reproduced the label in 11 of the 12 and left the twelfth unresolved, so that the advantage is claimed for those 11 alone. The margin is the mean of two states of a process in those sessions, in one of which the variant is the slower. It is larger than the movement between the two sessions of the median split's own net time and of the difference itself, yet three labels of this engine differ between them, and it is not presented as a practical performance advantage. That variant was a secondary engine of a second campaign, measured on one machine, in one arm and beside five engines rather than fourteen and the object count at which its sign changes depends on those conditions. The result therefore confines the negative finding against the object median to the submitted rule and to the one-axis rule without a cut-off and it does not establish an object count above which a binned heuristic repays. In a third campaign, a split at the midpoint of the node's box was faster than the object median in every cell of every arm, by 21 to 39 percent in arm B as implemented with the library's selection and, descriptively, by 4 to 8 percent there with construction set equal and no heuristic tested, as built, was faster than it in any cell. Through two campaigns, the heuristics were therefore measured against the slower of two inexpensive rules as implemented and the advantage of the variant with a cut-off is one over the object median alone. In a fourth campaign, one session, a hierarchy built in Morton order was faster in net time than the object median in every cell and than the midpoint split at 100 objects or more, the variant with a cut-off was labelled faster than it in none and was slower than the object median in 2 of its 11 claimed configurations. Every conclusion is confined to full rebuilding, box proxies, synthetic scenes and a single machine, a known limitation and none is claimed or excluded beyond 2,000 objects. It withdraws the recommendation, the ratios against brute force and the principal metric of the submitted version.

ACKNOWLEDGEMENTS

This work has been supported by Universiti Teknikal Malaysia Melaka. The authors thank the Centre for Research and Innovation Management, UTeM, for its continued support. The authors used large language models (Claude, Anthropic) throughout this revision: to write, test and run the measurement harness and the analysis, to draft the registration documents, the text of the manuscript and the response to the reviewers, to review these adversarially and to check each reference against its source. The authors directed the study, approved its design and criteria and take full responsibility for the content. Every reported value is computed by script from the data supplied with this submission. The authors declare no competing interests.

REFERENCES

[1] R. Weller, "A Brief Overview of Collision Detection," New Geometric Data Structures for Collision Detection and Haptics, pp. 9-46, DOI: 10.1007/978-3-319-01020-5_2, Cham, Springer, 2013.

[2] Y. R. Serpa and M. A. F. Rodrigues, "Broadmark: A Testing Framework for Broad-phase Collision Detection Algorithms," Computer Graphics Forum, vol. 39, no. 1, pp. 436-449, 10.1111/cgf.13884, 2020.

[3] I. Wald, "On Fast Construction of SAH-based Bounding Volume Hierarchies," Proc. of the IEEE

Symposium on Interactive Ray Tracing, pp. 33-40, DOI: 10.1109/RT.2007.4342588, 2007.

[4] J. T. Klosowski et al., "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," IEEE Trans. on Visualization and Comp. Graphics, vol. 4, no. 1, pp. 21-36, 10.1109/2945.675649, 1998.

[5] R. Ytterlid and E. Shellshear, "BVH Split Strategies for Fast Distance Queries," Journal of Computer Graphics Techniques, vol. 4, no. 1, pp. 1-25, 2015.

[6] T. Karras, "Maximizing Parallelism in the Construction of BVHs, Octrees and k-d Trees," Proc. of the Euro-Graphics/ACM SIGGRAPH Symposium on High Performance Graphics, pp. 33-37, DOI: 10.2312/EGGH/HPG12/033-037, 2012.

[7] X. Wang, M. Tang, D. Manocha and R. Tong, "Efficient BVH-based Collision Detection Scheme with Ordering and Restructuring," Comp. Graph. Forum, vol. 37, no. 2, pp. 227-237, 10.1111/cgf.13356, 2018.

[8] J. D. Cohen, M. C. Lin, D. Manocha and M. Ponamgi, "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-scale Environments," Proc. of the 1995 Symposium on Interactive 3D Graphics, pp. 189-196, DOI: 10.1145/199404.199437, 1995.

[9] D. J. Tracy, S. R. Buss and B. M. Woods, "Efficient Large-scale Sweep and Prune Methods with AABB Insertion and Removal," Proc. of the IEEE Virtual Reality Conf., pp. 191-198, DOI: 10.1109/VR.2009.4811022, 2009.

[10] J. Cao and M. Wang, "A Fast and Generalized Broad-phase Collision Detection Method Based on KDtree Spatial Subdivision and Sweep-and-prune," IEEE Access, vol. 11, pp. 44696-44710, 10.1109/ACCESS.2023.3274202, 2023.

[11] R. Lubbe et al., "Analysis of Parallel Spatial Partitioning Algorithms for GPU Based DEM," Computers and Geotechnics, vol. 125, Art. 103708, DOI: 10.1016/j.compgeo.2020.103708, 2020.

[12] N.-J. Sung and M. Hong, "Toward Real-time Scalable Rigid-body Simulation Using GPU-optimized Collision Detection and Response," Mathematics, vol. 13, no. 19, Art. 3230, DOI: 10.3390/math13193230, 2025.

[13] G. van den Bergen, "Efficient Collision Detection of Complex Deformable Models Using AABB Trees," J. of Graphics Tools, vol. 2, no. 4, pp. 1-13, DOI: 10.1080/10867651.1997.10487480, 1997.

[14] G. Zachmann, "Minimal Hierarchical Collision Detection," Proc. of the ACM Symposium on Virtual Reality Software and Technology, pp. 121-128, DOI: 10.1145/585740.585761, 2002.

[15] H. A. Sulaiman and A. Bade, "Bounding Volume Hierarchies for Collision Detection," in Computer Graphics, N. Mukai, Ed., Rijeka, Croatia: InTech, pp. 39-54, DOI: 10.5772/35555, 2012.

[16] D. J. MacDonald and K. S. Booth, "Heuristics for Ray Tracing Using Space Subdivision," The Visual Computer, vol. 6, no. 3, pp. 153-166, DOI: 10.1007/BF01911006, 1990.

[17] I. Wald, S. Boulos and P. Shirley, "Ray Tracing Deformable Scenes Using Dynamic Bounding Volume Hierarchies," ACM Trans. on Graphics, vol. 26, no. 1, Art. 6, DOI: 10.1145/1189762.1206075, 2007.

[18] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke and D. Manocha, "Fast BVH Construction on GPUs," Computer Graphics Forum, vol. 28, no. 2, pp. 375-384, DOI: 10.1111/j.1467-8659.2009.01377.x, 2009.

[19] T. Karras and T. Aila, "Fast Parallel Construction of High-quality Bounding Volume Hierarchies," Proc. of the 5th High-Performance Graphics Conference, pp. 89-99, DOI: 10.1145/2492045.2492055, 2013.

[20] C. Benthin et al., "PLOC++: Parallel Locally-ordered Clustering for Bounding Volume Hierarchy Construction Revisited," Proc. of the ACM on Computer Graphics and Interactive Techniques, vol. 5, no. 3, pp. 1-13, DOI: 10.1145/3543867, 2022.

[21] C. Benthin et al., "H-PLOC: Hierarchical Parallel Locally-ordered Clustering for Bounding Volume Hierarchy Construction," Proc. of the ACM on Computer Graphics and Interactive Techniques, vol. 7, no. 3, pp. 1-14, DOI: 10.1145/3675377, 2024.

[22] D. Meister et al., "A Survey on Bounding Volume Hierarchies for Ray Tracing," Computer Graphics Forum, vol. 40, no. 2, pp. 683-712, DOI: 10.1111/cgf.142662, 2021.

[23] N. Vitsas, I. Evangelou, G. Papaioannou and A. Gkaravelis, "Parallel Transformation of Bounding Volume Hierarchies into Oriented Bounding Box Trees," Computer Graphics Forum, vol. 42, no. 2, pp. 245-254, DOI: 10.1111/cgf.14758, 2023.

[24] M. Káčerik and J. Bittner, "SAH-optimized k-DOP Hierarchies for Ray Tracing," Proc. of the ACM on Computer Graphics and Interactive Techniques, vol. 7, no. 3, pp. 1-16, DOI: 10.1145/3675391, 2024.

[25] M. Káčerik and J. Bittner, "SOBB: Skewed Oriented Bounding Boxes for Ray Tracing," Computer Graphics Forum, vol. 44, no. 2, Art. e70062, DOI: 10.1111/cgf.70062, 2025.

[26] F. M. Chitalu, C. Dubach and T. Komura, "Binary Ostensibly-implicit Trees for Fast Collision Detection," Computer Graphics Forum, vol. 39, no. 2, pp. 509-521, DOI: 10.1111/cgf.13948, 2020.

[27] H. A. Sulaiman, M. L. Dolhalit, N. Abdullasim, I. A. Zolkifly and A. Bade, "SAH *versus* Morton-code BVH Construction for Real-time Broad-phase Collision Detection: A Controlled CPU Benchmark," JOIV: Int. J. on Informatics Visualization, vol. 11, no. 1, Early Publishing, 2027.

[28] D. K. Mandarapu et al., "Rethinking Collision Detection on GPU Ray Tracing Architecture," Proc. of the 40th ACM Int. Conf. on Supercomputing, pp. 960-971, DOI: 10.1145/3797905.3807836, 2026.

[29] S. Koh, Y. Kim and I. Ihm, "Ray Tracing-based Real-time Collision Detection Using Bounding Mesh of

Polygonal Model," Journal of KIISE, vol. 51, no. 2, pp. 173-183, DOI: 10.5626/jok.2024.51.2.173, 2024.
[30] P. Fan, W. Wang, R. Tong, H. Li and M. Tang, "gDist: Efficient Distance Computation Between 3D Meshes on GPU," SIGGRAPH Asia 2024 Conf. Papers, pp. 1-11, DOI: 10.1145/3680528.3687619, 2024.
[31] J. Pelletier-Guénette et al., "Real-time Triangle-SDF Continuous Collision Detection," Proc. of the ACM on Computer Graphics and Interactive Techniques, vol. 8, no. 4, pp. 1-22, DOI: 10.1145/3747862, 2025.
[32] A. Yazdani and A. Wachs, "Performance Optimization of GJK Collision Detection in Discrete Element Simulations," Computer Physics Communi., vol. 316, Art. 109768, 10.1016/j.cpc.2025.109768, 2025.
[33] A. Motte, C. Bovet, V. Chiaruttini, O. Jamond and B. Prabel, "Massively Parallel Tandem Traversal of Bounding Volume Hierarchies for Geometric Queries on Distributed Meshes," Computers and Structures, vol. 316, Art. 107843, DOI: 10.1016/j.compstruc.2025.107843, 2025.
[34] B. Wang et al., "A Large-scale Benchmark and an Inclusion-based Algorithm for Continuous Collision Detection," ACM Trans. on Graphics, vol. 40, no. 5, pp. 1-16, DOI: 10.1145/3460775, 2021.
[35] C. Ericson, Real-time Collision Detection, ISBN-13: 9781558607323, San Francisco, USA: Morgan Kaufmann, 2005.
[36] T. Larsson and T. Akenine-Möller, "A Dynamic Bounding Volume Hierarchy for Generalized Collision Detection," Computers and Graphics, vol. 30, no. 3, pp. 450-459, DOI: 10.1016/j.cag.2006.02.011, 2006.
[37] K. Shoemake, "Uniform Random Rotations," Graphics Gems III, Elsevier, pp. 124-132, DOI: 10.1016/B978-0-08-050755-2.50036-1, 1992.
[38] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator," ACM Trans. on Modeling and Computer Simulation, vol. 8, no. 1, pp. 3-30, DOI: 10.1145/272991.272995, 1998.
[39] J. Durbin and G. S. Watson, "Testing for Serial Correlation in Least Squares Regression: I," Biometrika, vol. 37, no. 3-4, pp. 409-428, DOI: 10.1093/biomet/37.3-4.409, 1950.
[40] S. S. Shapiro and M. B. Wilk, "An Analysis of Variance Test for Normality (complete samples)," Biometrika, vol. 52, no. 3-4, pp. 591-611, DOI: 10.1093/biomet/52.3-4.591, 1965.
[41] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," The Annals of Statistics, vol. 7, no. 1, pp. 1-26, DOI: 10.1214/aos/1176344552, 1979.
[42] F. Wilcoxon, "Individual Comparisons by Ranking Methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80-83, DOI: 10.2307/3001968, 1945.
[43] S. Holm, "A Simple Sequentially Rejective Multiple Test Procedure," Scandinavian Journal of Statistics, vol. 6, pp. 65-70, 1979.
[44] A. Bade et al., "New Bounding Volume Approach for Detecting Collision in Computer Animation," Malaysian J. of Computer Science, vol. 19, no. 2, pp. 127-140, 2006.

ملخص البحث:

في سياق الهياكل الرقمية للمرحلة الأوّلية الّتي يُعاد بناؤها في كلّ إطار، وُجد أنّ أسلوب "الاستدلال القائم على مساحة السّطح" المعتمد على المحاور الثلاثة يجري اختبارات صناديق أقلّ بنسبة تتراوح بين 17.8% و 46.0% مقارنةً مع أسلوب "التّقسيم عند الوسيط"، ومع ذلك، فإنّه يستغرق وقتاً إجمالياً أطول بنسبة تتراوح بين 9% و 365% عند التّعامل مع ما يصل إلى 2000 كائن (مع عدم استبعاد إمكانية حدوث تقاطع في النّتائج عند أعدادٍ أكبر). وينطبق الأمر ذاته على أسلوب الاستدلال المعتمد على محور واحد. أمّا عند استخدام "التّقسيم عند الوسيط" مع ثمانية كائنات أو أقلّ، فقد تفوّق هذا الأسلوب بنسبة تتراوح بين 1.1% و 2.4% في 12 إعداداً مختلفاً، وتكرّرت هذه النّتيجة في 11 إعداداً عند إعادة التّجربة وفي 10 إعداداتٍ في جولةٍ أخرى، وهي فوارق ضئيلة جدّاً لا تكفي للجزم بتفوّق أحد الأسلوبين بشكلٍ قاطع. وكان تقسيم المربّع عند منتصفه أسرع من متوسّط الكائن في جميع التّكوينات الثّلاثين ذات التّحكم في الكثافة بنسبة تتراوح بين 21% و 39% عند تطبيقه.

وكان التّسلسل الهرمي المبني وفقاً لترتيب مورتون، والّذي تمّ قياسه بناءً على طلب أحد المحكّمين، أسرع من كليهما عند 100 كائن أو أكثر، وخاصّة في مرحلة الإنشاء. ولم تكن أيّ طريقةٍ استدلالية أسرع من تقسيم المنتصف، ولا أيّ من المتغيّرات أسرع من بناء مورتون.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).