Constructing Efficient Mesh-Based Global Grid Systems with Reduced Distortions

Lakin Wecker, John Hall, Faramarz Samavati

APPENDIX

A. Performance Benchmarks Methodology

Our experimental benchmarks were run using the `nanobench` library [2]. Our DGGS implementation and the benchmarks are written in C++20 and compiled with GCC 12.3.0 on Ubuntu 22.04 and using `Eigen 3` [1]. The benchmarks were all run on a 13th-generation Intel Core i9-13900K. `Systemd` was configured to avoid scheduling other processes on a particular CPU core. All benchmarks were isolated to this same core using TASKSET program, and all benchmarks were run on a single thread. The benchmarks are run on every 2nd level of resolution starting at level 0 and ending on resolution level 24 except for the PARENT operation, which skipped resolution level 0 as there is no parent at that level. We chose these levels of resolution to be a good representation of the various resolution levels available in both MBD and the DGGS we compare against. For mesh-based DGGS, we report the speed results for $\mathbf{PM}_s$, $\mathbf{PRM}_s$, $\mathbf{PR}^2\mathbf{M}_s$, and $\mathbf{PR}^3\mathbf{M}_s$.

To determine the expected performance of the operations for all regions of the DGGS, we randomly sample 10,000 points across the sphere. Then, each operation is run as a nanobench benchmark for each point at every resolution, and the results are stored in a CSV file. The following tables are all generated by processing these results in the R programming language to calculate the mean, standard deviation, and 90% quantile. The following figures are also generated by processing these results in the R programming language using the tidyverse [4] and ggstatsplot [3] libraries.

B. Memory Usage Statistics

We discuss how much memory is used in our implementation of MBD in the appendix (appendix C). The primary takeaway from this discussion is that the hash-encoding lookup structure represents the bulk of the memory usage within our DGGS and that its size is dependent on the number of buckets. The number of buckets depends on the heuristic we use to choose how many times we should partition the latitude and longitude ranges for the partitioning structure. This parameter can be modified to control the memory usage of the hash-encoding lookup structure. In table I, we use a factor of 0.1 and report on the number of partitions, the size of the hash-encoding lookup structure, and the size of the resulting DGGS. We conclude that all of the suggested DGGS fit within the cache size of modern CPU architectures, and thus, MBD is highly efficient.

L. Wecker, J. Hall and F. F. Samavati are with the University of Calgary

TABLE S1: The memory usage of our recommended base polyhedra. As the resolution increases, so does the number of partitions we use for latitude (ϕ) and longitude (θ). The hash-encoding lookup structure represents most of the memory usage, as expected. Each level of refinement introduces four times more faces, which increases memory usage by approximately four times as well. $\vee$ indicates min, $\approx$ indicates median, $\wedge$ indicates max.

	Lookup Grid		Mem. use (MB)		Cells per bucket
P_b	ϕ	$\theta \uparrow$	Lookup	Total	$\vee \mid \approx \mid \wedge$
P	69	137	0.13	0.14	1 | 1.25 | 6
$\mathbf{PRM}_s$	139	277	0.51	0.58	1 | 1.29 | 6
$\mathbf{PR}^2\mathbf{M}_s$	278	555	2.06	2.32	1 | 1.27 | 6
$\mathbf{PR}^3\mathbf{M}_s$	555	1109	8.22	9.24	1 | 1.25 | 6

C. Memory Usage Discussion

DGGS are comprised of the faces of the base polyhedron, cached information for efficient operations, and the hash-encoding lookup structure. The mesh's geometry is stored as a single array of vertices, where every vertex contains three double-floating point quantities. Given n faces in the base polyhedron, this results in $n * 8 * 3$ for the geometry of the base polyhedron.

In addition, we store another array of per-face pre-calculated information that aids in the calculation of certain operations:

- **Projection quantities**: A vector representing the normal of the triangle (pointing away from the origin) and the scalar dot product between the normal and the first vertex of the face. *These are used when projecting or inverse-projecting between the spherical and planar counterparts.*
- **Point in cell quantities**: 3 vectors representing the normals of the planes formed by each edge, pointing inwards towards the triangle. *These are used for determining if a point is contained within the triangle when more than one candidate face is returned from the hash-encoding lookup structure.*
- **Information for calculating barycentric coordinates**: Three vectors and four scalar quantities. *These are used when calculating the barycentric coordinates of an incoming point and are quantities that depend only on the triangle and thus can be precalculated to improve the speed of this operation.*

In total, $n * 208$ bytes of information are stored to improve the performance of various operations.

To allow for the neighbor operation to be calculated when neighbors cross the edges of a face of the base polyhedron, we

also store connectivity between each face. These are stored in a format that maps each face i and each edge $j \in [0,2]$ tuple (i, j) to face k that is a neighbor of i across edge j. In addition, a second lookup is provided that maps each (i, j) to m where $m \in [0,2]$ is the edge number for face k, which shares an edge with face i. Thus, the neighbor of face i across edge j is k, and the neighbor face k across edge m is i.

The hash encoding lookup structure has the largest memory footprint in the DGGS. Recall that we use a heuristic to choose how many times we should partition the latitude and longitude ranges for the partitioning structure. The bucket edge size is configurable, representing a tradeoff between performance and memory usage. However, all of our experimental evaluations have been done with a value of 0.1 In the table I, we report how many partitions are used with this factor for the latitude (denoted a) and longitude (denoted b).

The hash encoding information is stored in a contiguous array to ensure to ensure values are in the CPU cache for its operations. We calculate this array in a two-pass pre-processing step, which uses two different representations, and only the second representation is maintained. In the first pass, we calculate which base polyhedra are candidates for each bucket in the lookup structure and store this in a dynamically allocated structure. We then calculate the maximum number of candidate faces that any bucket contains, denoted e. In a second pass, we convert this structure to a single contiguous array of 16-bit unsigned integers (u_{16}). The array contains enough space for $a*b*(e+1)$ values of type u_{16}. Each bucket has $e+1$ values, the first in the sequence representing the number of candidate faces and each of the rest representing the index of a candidate face. Thus, the number of bytes used by the overall structure is $a*b*(e+1)*2$ for the array and 24 for storing each of a, b, and e.

Table table I reports the memory usage for our recommended base polyhedra.

D. Other Operation results and discussion

The results for the CHILDREN and PARENT operations are in **??** and the associated graphs are in the supplementary material.

The table and graph for both of these operations show that they are highly efficient and run in constant time even when operating at high resolutions, and even when the base polyhedron is high-resolution. The results for the NEIGHBOR operation (Section **??**) are in **??** and the graphs are in the supplementary material. The graph and for this operation demonstrate that there are three cases, each of which is efficient and constant time. The three cases are:

1) the cell has no neighbors in an adjacent base-cell,
2) the cell has one neighbor in an adjacent base-cell,
3) the cell has two neighbors in an adjacent base-cell.

Each case is constant time, with each subsequent neighbor across a base-cell adding a constant amount of time to the operation. Also, we note that the expected number of cells in each category changes as we increase resolutions. The number of those with no neighbors in a neighboring base-cell will increase more quickly than the other two, and the graphs confirm that this category of points dominates at high resolutions. Thus, the expected runtime of the NEIGHBOR operation decreases as we increase resolutions. This table and these graphs also confirm that the NEIGHBOR operation is constant time even when using a high-resolution base polyhedron.

The SPHERICALGEOMETRYFROMINDEX operation results are **??** and graphed in the supplementary material. The table and graph for this operation show that it is constant time, even when operating at high resolutions and even when the base polyhedron is high-resolution.

All of these graphs and tables show that MBD operations have extremely efficient constant time operations. This efficiency continues to be present even with dealing with high-resolution data or using a high resolution base polyhedron.

E. Neighbor ACM Index Swap

If we denote the edges 0, 1, and 2 for $\overline{\mathbf{v}_A\mathbf{v}_B}$, $\overline{\mathbf{v}_B\mathbf{v}_C}$, and $\overline{\mathbf{v}_C\mathbf{v}_A}$ respectively and we sum the connected edges modulo 3, you'll notice that we obtain the same answer for each scenario in the same group (fig. 1a). Thus, we simply need to find an equation that efficiently maps $0 \to 0$, $2 \to 1$, and $1 \to 2$. This can be done via a straightforward if statement or the following bitwise operations. Assuming $\vee$ is bitwise or, $\wedge$ is bitwise and, $\oplus$ is bitwise exclusive or, $\gg$ is shifting bits towards the least significant bit, and $\ll$ is shifting bits towards the most significant bit, and j is the sum of the edge indices, modulo 3, then the operations are: $x = (j \gg 1) \oplus (j \wedge 1)$ and $j = j \oplus ((x \ll 1) \vee x)$. We swap the bits in position j and $j+1$ modulo 3.

e_1	e_2	Indices	(e_1+e_2) mod 3	First
0	0	$a \leftrightarrow b$	0	0
1	2	$a \leftrightarrow b$	0	0
2	1	$a \leftrightarrow b$	0	0
1	1	$b \leftrightarrow c$	2	1
0	2	$b \leftrightarrow c$	2	1
2	0	$b \leftrightarrow c$	2	1
0	1	$c \leftrightarrow a$	1	2
1	0	$c \leftrightarrow a$	1	2
2	2	$c \leftrightarrow a$	1	2

(a) If we number the edges and sum the edge number for each of the cells modulo 3, then we get the same value for each edge combination that needs the same swap.

F. Figures and Tables

REFERENCES

[1] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[2] M. Leitner-Ankerl. nanobench documentation, 2023. (accessed on 28 Aug 2023).

[3] I. Patil. Visualizations with statistical details: The 'ggstatsplot' approach. *Journal of Open Source Software*, 6(61):3167, 2021. doi: 10.21105/joss.03167

TABLE S2: The statistics of the angle distortion for our proposed P_b. Note that the angle distortion is heavily correlated with the number of faces with a small correlation to the compactness of the faces.

			Angle distortion (d_θ)				
	P_b	# Cells	Mean	Std. Dev.	Min	Max	RMSE ↑
1	$\mathbf{PR}^4\mathbf{M}_s$	15360	0.0001519	0.0000126	0.000000067	0.0003172	0.0001525
2	$\mathbf{PR}^4\mathbf{M}_g$	15360	0.0001581	0.0000137	0.000000014	0.0003323	0.0001587
3	$\mathbf{DR}^3\mathbf{M}_s$	7680	0.0004517	0.0000355	0.000000333	0.0009453	0.0004531
4	$\mathbf{IR}^4\mathbf{M}_s$	5120	0.0004584	0.0000437	0.000000325	0.0009517	0.0004605
5	$\mathbf{I}(\mathbf{RM}_g)^4$	5120	0.0005104	0.0000594	0.000000293	0.0010456	0.0005139
6	$\mathbf{IR}^4\mathbf{M}_g$	5120	0.0005122	0.0000634	0.000000256	0.0010613	0.0005161
7	$\mathbf{PR}^3\mathbf{M}_s$	3840	0.0006000	0.0000708	0.000000004	0.0012511	0.0006042
8	$\mathbf{PR}^3\mathbf{M}_g$	3840	0.0006244	0.0000714	0.000000307	0.0012875	0.0006285
9	$\mathbf{DR}^2\mathbf{M}_s$	1920	0.0017502	0.0002073	0.000000473	0.0036537	0.0017624
10	$\mathbf{IR}^3\mathbf{M}_s$	1280	0.0018034	0.0002665	0.000003181	0.0039120	0.0018230
11	$\mathbf{I}(\mathbf{RM}_g)^3$	1280	0.0019766	0.0003110	0.000001635	0.0042284	0.0020009
12	$\mathbf{IR}^3\mathbf{M}_g$	1280	0.0019846	0.0003167	0.000001696	0.0042283	0.0020097
13	$\mathbf{PR}^2\mathbf{M}_s$	960	0.0023519	0.0003769	0.000000981	0.0051138	0.0023820
14	$\mathbf{PR}^2\mathbf{M}_g$	960	0.0024304	0.0003707	0.000000195	0.0052079	0.0024585
15	$\mathbf{DRM}_s$	480	0.0067394	0.0011327	0.000010419	0.0143672	0.0068339
16	$\mathbf{IR}^2\mathbf{M}_s$	320	0.0070195	0.0014577	0.000000385	0.0157445	0.0071693
17	$\mathbf{I}(\mathbf{RM}_g)^2$	320	0.0075153	0.0015660	0.000013065	0.0169659	0.0076767
18	$\mathbf{IR}^2\mathbf{M}_g$	320	0.0075383	0.0015622	0.000018540	0.0169661	0.0076984
19	$\mathbf{PRM}_s$	240	0.0090826	0.0020021	0.000007416	0.0207418	0.0093006
20	$\mathbf{PRM}_g$	240	0.0091206	0.0020043	0.000020904	0.0197101	0.0093382
21	$\mathbf{D}$	120	0.0239404	0.0059065	0.000075524	0.0529297	0.0246581
22	$\mathbf{IRM}_s$	80	0.0266095	0.0077931	0.000000957	0.0662213	0.0277271
23	$\mathbf{IRM}_g$	80	0.0266095	0.0077931	0.000000958	0.0662215	0.0277271
24	$\mathbf{P}$	60	0.0325684	0.0101923	0.000500642	0.0763061	0.0341256
25	$\mathbf{I}$	20	0.0874756	0.0353505	0.000397421	0.2190605	0.0943469

[4] H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686

TABLE S3: Statistics regarding the total area distortion in each of our P_b due to the initial area non-uniformity and the projection.

			Total area distortion (d_A)				
	P_b	# Cells	Mean	Std. Dev.	Min	Max	RMSE ↑
1	$\mathbf{PR}^4\mathbf{M}_s$	15360	1.0	0.00663512	0.9321378	1.03389988	0.00663512
2	$\mathbf{DR}^3\mathbf{M}_s$	7680	1.0	0.00719718	0.9797803	1.02153464	0.00719718
3	$\mathbf{DR}^2\mathbf{M}_s$	1920	1.0	0.01042000	0.9779221	1.02248567	0.01042000
4	$\mathbf{PR}^3\mathbf{M}_s$	3840	1.0	0.01205686	0.9311533	1.03526341	0.01205685
5	$\mathbf{DRM}_s$	480	1.0	0.01362017	0.9627823	1.02592901	0.01362016
6	$\mathbf{IR}^4\mathbf{M}_s$	5120	1.0	0.01858054	0.9033260	1.06316711	0.01858053
7	$\mathbf{PR}^2\mathbf{M}_s$	960	1.0	0.02144050	0.9272125	1.04076141	0.02144049
8	$\mathbf{PRM}_g$	240	1.0	0.03009671	0.9354920	1.06407083	0.03009667
9	$\mathbf{IR}^3\mathbf{M}_s$	1280	1.0	0.03225850	0.9005473	1.07744319	0.03225849
10	$\mathbf{PRM}_s$	240	1.0	0.03649345	0.9114017	1.06349944	0.03649339
11	$\mathbf{D}$	120	1.0	0.03686068	0.9016534	1.05412000	0.03686057
12	$\mathbf{PR}^2\mathbf{M}_g$	960	1.0	0.04277345	0.9072334	1.06380043	0.04277344
13	$\mathbf{PR}^3\mathbf{M}_g$	3840	1.0	0.04629806	0.8762423	1.06370923	0.04629805
14	$\mathbf{PR}^4\mathbf{M}_g$	15360	1.0	0.04719103	0.8573973	1.06533280	0.04719103
15	$\mathbf{P}$	60	1.0	0.04720455	0.8473916	1.06538929	0.04720426
16	$\mathbf{IR}^2\mathbf{M}_s$	320	1.0	0.05517632	0.8894081	1.08643307	0.05517626
17	$\mathbf{I}(\mathbf{RM}_g)^4$	5120	1.0	0.08644592	0.9263581	1.20656602	0.08644591
18	$\mathbf{I}(\mathbf{RM}_g)^3$	1280	1.0	0.08646478	0.9223354	1.20656300	0.08646475
19	$\mathbf{I}(\mathbf{RM}_g)^2$	320	1.0	0.08676194	0.9063684	1.20655091	0.08676184
20	$\mathbf{IRM}_g$	80	1.0	0.09114404	0.8444595	1.20650257	0.09114363
21	$\mathbf{IRM}_s$	80	1.0	0.09114404	0.8444595	1.20650257	0.09114363
22	$\mathbf{IR}^2\mathbf{M}_g$	320	1.0	0.12040402	0.7652965	1.20655091	0.12040388
23	$\mathbf{IR}^3\mathbf{M}_g$	1280	1.0	0.13012447	0.6890477	1.20656300	0.13012443
24	$\mathbf{IR}^4\mathbf{M}_g$	5120	1.0	0.13259216	0.6474662	1.20656602	0.13259215
25	$\mathbf{I}$	20	1.0	0.13263882	0.6266417	1.20630924	0.13263639

TABLE S4: Statistics regarding the area non-uniformity caused by P_b.

			Area Non-uniformity (d_A) in the base polyhedron				
	P_b	# Cells	Mean	Std. Dev.	Min	Max	RMSE ↑
1	$\mathbf{D}$	120	1.0	0.0	1.0	1.0	0.0
2	$\mathbf{I}$	20	1.0	0.0	1.0	1.0	0.0
3	$\mathbf{P}$	60	1.0	0.0	1.0	1.0	0.0
4	$\mathbf{PR}^4\mathbf{M}_s$	15360	1.0	0.00663269	0.9327243	1.03364117	0.00663248
5	$\mathbf{DR}^3\mathbf{M}_s$	7680	1.0	0.00717223	0.9814637	1.02057298	0.00717176
6	$\mathbf{DRM}_s$	480	1.0	0.00969423	0.9873097	1.01049364	0.00968413
7	$\mathbf{DR}^2\mathbf{M}_s$	1920	1.0	0.01013952	0.9840500	1.01863634	0.01013688
8	$\mathbf{PR}^3\mathbf{M}_s$	3840	1.0	0.01203511	0.9335000	1.03422711	0.01203355
9	$\mathbf{IR}^4\mathbf{M}_s$	5120	1.0	0.01857384	0.9049809	1.06236385	0.01857203
10	$\mathbf{PR}^2\mathbf{M}_s$	960	1.0	0.02124141	0.9366072	1.03659233	0.02123035
11	$\mathbf{PRM}_g$	240	1.0	0.02765129	0.9752723	1.04680294	0.02759363
12	$\mathbf{IR}^3\mathbf{M}_s$	1280	1.0	0.03219270	0.9071713	1.07416969	0.03218012
13	$\mathbf{PRM}_s$	240	1.0	0.03455178	0.9491076	1.04642628	0.03447972
14	$\mathbf{PR}^2\mathbf{M}_g$	960	1.0	0.04268901	0.9162252	1.05938730	0.04266677
15	$\mathbf{PR}^3\mathbf{M}_g$	3840	1.0	0.04629791	0.8783200	1.06260404	0.04629188
16	$\mathbf{PR}^4\mathbf{M}_g$	15360	1.0	0.04719219	0.8578935	1.06505522	0.04719065
17	$\mathbf{IR}^2\mathbf{M}_s$	320	1.0	0.05452483	0.9159699	1.07325477	0.05443957
18	$\mathbf{IRM}_g$	80	1.0	0.08423406	0.9516724	1.14498294	0.08370594
19	$\mathbf{IRM}_s$	80	1.0	0.08423406	0.9516724	1.14498294	0.08370594
20	$\mathbf{I}(\mathbf{RM}_g)^2$	320	1.0	0.08641395	0.9339726	1.19035818	0.08627883
21	$\mathbf{I}(\mathbf{RM}_g)^3$	1280	1.0	0.08646814	0.9292853	1.20245990	0.08643435
22	$\mathbf{I}(\mathbf{RM}_g)^4$	5120	1.0	0.08645245	0.9280986	1.20553674	0.08644401
23	$\mathbf{IR}^2\mathbf{M}_g$	320	1.0	0.12024577	0.7848607	1.19035818	0.12005774
24	$\mathbf{IR}^3\mathbf{M}_g$	1280	1.0	0.13015502	0.6929174	1.20245990	0.13010417
25	$\mathbf{IR}^4\mathbf{M}_g$	5120	1.0	0.13260386	0.6483158	1.20553674	0.13259091

TABLE 5: Statistics regarding the Area distortion that is caused by the projection itself. Note that it is primarily correlated to the number of base-cells. A notable exception is the **D** where the low compactness at resolution 0 increases the distortion.

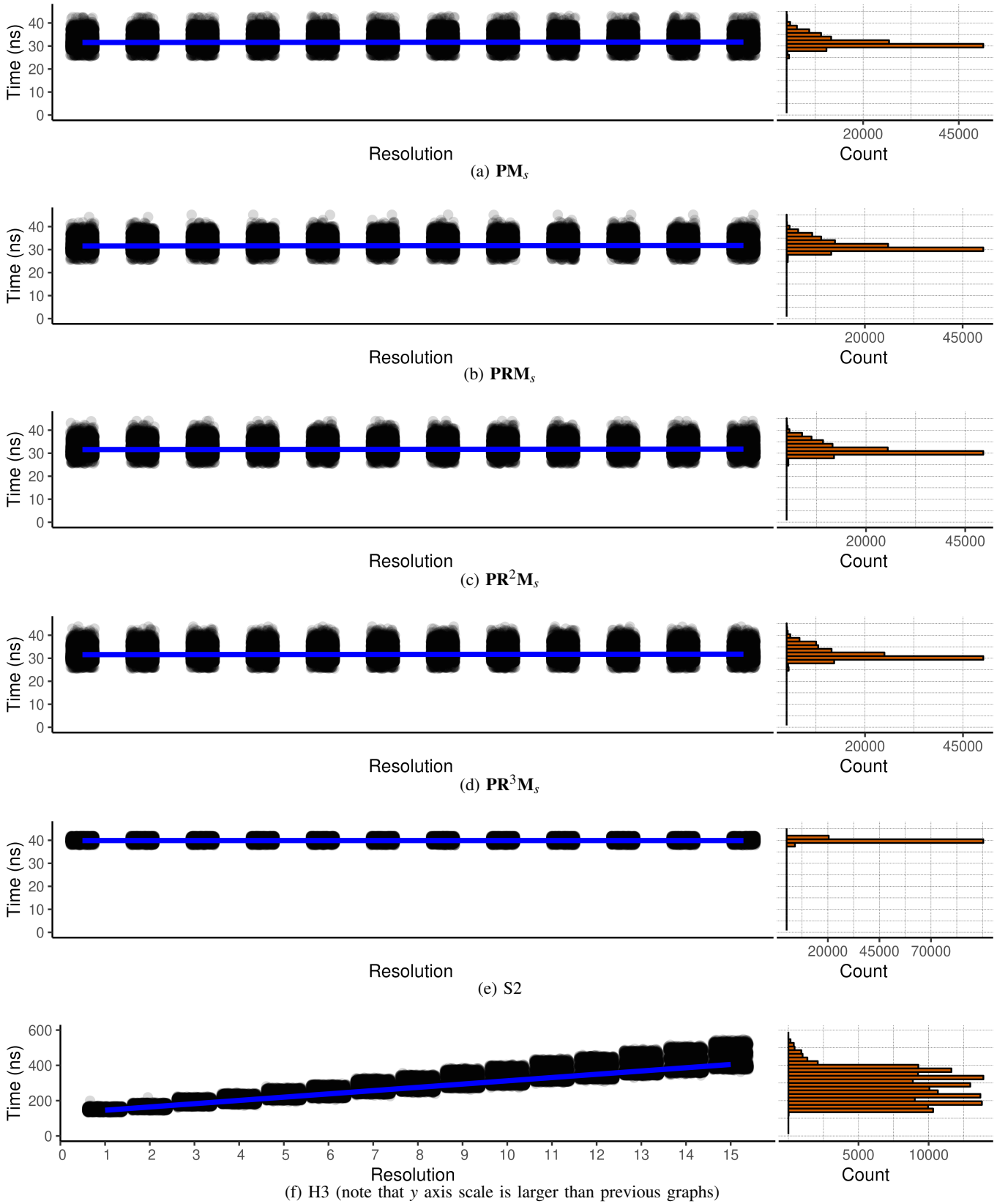
			Area Distortion (d_A) due to projection				
	P_b	# Cells	Mean	Std. Dev.	Min	Max	RMSE ↑
1	$\mathbf{PR}^4\mathbf{M}_s$	15360	1.0	0.00018740	0.9993186	1.00025174	0.00018740
2	$\mathbf{PR}^4\mathbf{M}_g$	15360	1.0	0.00018801	0.9992951	1.00026063	0.00018801
3	$\mathbf{IR}^4\mathbf{M}_g$	5120	1.0	0.00055405	0.9976505	1.00085379	0.00055405
4	$\mathbf{IR}^4\mathbf{M}_s$	5120	1.0	0.00056177	0.9979180	1.00075610	0.00056177
5	$\mathbf{I}(\mathbf{RM}_g)^4$	5120	1.0	0.00056444	0.9976505	1.00085379	0.00056444
6	$\mathbf{DR}^3\mathbf{M}_s$	7680	1.0	0.00060418	0.9982376	1.00104192	0.00060418
7	$\mathbf{PR}^3\mathbf{M}_s$	3840	1.0	0.00074917	0.9972767	1.00100359	0.00074917
8	$\mathbf{PR}^3\mathbf{M}_g$	3840	1.0	0.00075195	0.9971912	1.00104008	0.00075195
9	$\mathbf{IR}^3\mathbf{M}_g$	1280	1.0	0.00221595	0.9906685	1.00341226	0.00221595
10	$\mathbf{IR}^3\mathbf{M}_s$	1280	1.0	0.00224309	0.9916587	1.00304746	0.00224309
11	$\mathbf{I}(\mathbf{RM}_g)^3$	1280	1.0	0.00225618	0.9906685	1.00341226	0.00225618
12	$\mathbf{DR}^2\mathbf{M}_s$	1920	1.0	0.00241156	0.9930414	1.00403638	0.00241156
13	$\mathbf{PR}^2\mathbf{M}_s$	960	1.0	0.00299247	0.9891434	1.00402191	0.00299247
14	$\mathbf{PR}^2\mathbf{M}_g$	960	1.0	0.00300617	0.9888503	1.00416574	0.00300617
15	$\mathbf{IR}^2\mathbf{M}_g$	320	1.0	0.00886019	0.9637018	1.01360324	0.00886018
16	$\mathbf{IR}^2\mathbf{M}_s$	320	1.0	0.00893648	0.9670541	1.01227882	0.00893647
17	$\mathbf{I}(\mathbf{RM}_g)^2$	320	1.0	0.00899873	0.9637018	1.01360324	0.00899872
18	$\mathbf{DRM}_s$	480	1.0	0.00956758	0.9728399	1.01550491	0.00956757
19	$\mathbf{PRM}_s$	240	1.0	0.01193254	0.9571432	1.01631569	0.01193252
20	$\mathbf{PRM}_g$	240	1.0	0.01199818	0.9569431	1.01649584	0.01199816
21	$\mathbf{IRM}_g$	80	1.0	0.03554681	0.8692450	1.05372973	0.03554665
22	$\mathbf{IRM}_s$	80	1.0	0.03554681	0.8692450	1.05372973	0.03554665
23	**D**	120	1.0	0.03686068	0.9016534	1.05412000	0.03686057
24	**P**	60	1.0	0.04720455	0.8473916	1.06538929	0.04720426
25	**I**	20	1.0	0.13263882	0.6266417	1.20630924	0.13263639

TABLE S6: The statistics of the compactness of the cells in our proposed P_b. The compactness is heavily correlated to the compactness of the original faces of the input polyhdron. Thus, the icosahedron generally produces the most compact faces, followed by the pentakis dodecahedron. The disdyakis triacontahedron is a distant last, with the least compact faces of any of them.

			Compactness (d_C) of our proposed DGGS				
	P_b	# Cells	Mean	Std. Dev.	Min	Max	RMSE ↑
1	**I**	20	0.9418833	0.001814312	0.9335715	0.9999999	0.05814499
2	$\mathbf{IR}^4\mathbf{M}_g$	5120	0.9417971	0.001722388	0.9335721	0.9432760	0.05822834
3	$\mathbf{IR}^3\mathbf{M}_g$	1280	0.9416372	0.002150317	0.9335729	0.9441464	0.05840242
4	$\mathbf{IR}^2\mathbf{M}_g$	320	0.9410083	0.003106135	0.9335745	0.9475639	0.05907345
5	$\mathbf{IR}^2\mathbf{M}_s$	320	0.9397088	0.002637709	0.9335770	0.9465663	0.06034883
6	$\mathbf{IR}^3\mathbf{M}_s$	1280	0.9395723	0.001886925	0.9335772	0.9440226	0.06045717
7	$\mathbf{IR}^4\mathbf{M}_s$	5120	0.9394540	0.001539682	0.9335772	0.9432118	0.06056555
8	$\mathbf{PR}^2\mathbf{M}_s$	960	0.9394006	0.001798091	0.9335654	0.9427323	0.06062607
9	$\mathbf{PR}^3\mathbf{M}_s$	3840	0.9393582	0.001494640	0.9335654	0.9421862	0.06066020
10	$\mathbf{PR}^4\mathbf{M}_s$	15360	0.9393230	0.001398864	0.9335654	0.9421495	0.06069315
11	$\mathbf{PRM}_s$	240	0.9392988	0.002400984	0.9335652	0.9459854	0.06074866
12	$\mathbf{IRM}_g$	80	0.9391254	0.002424679	0.9335760	0.9602923	0.06092283
13	$\mathbf{IRM}_s$	80	0.9391254	0.002424679	0.9335760	0.9602923	0.06092283
14	**P**	60	0.9390448	0.001208531	0.9335648	0.9589797	0.06096719
15	$\mathbf{PR}^4\mathbf{M}_g$	15360	0.9390248	0.001188993	0.9335649	0.9409994	0.06098676
16	$\mathbf{PR}^3\mathbf{M}_g$	3840	0.9390071	0.001265904	0.9335650	0.9410292	0.06100604
17	$\mathbf{PR}^2\mathbf{M}_g$	960	0.9389306	0.001516319	0.9335652	0.9415958	0.06108821
18	$\mathbf{I}(\mathbf{RM}_g)^2$	320	0.9389177	0.002636001	0.9335773	0.9475639	0.06113912
19	$\mathbf{I}(\mathbf{RM}_g)^3$	1280	0.9388943	0.002642109	0.9335776	0.9441464	0.06116278
20	$\mathbf{I}(\mathbf{RM}_g)^4$	5120	0.9388828	0.002640188	0.9335777	0.9432760	0.06117422
21	$\mathbf{PRM}_g$	240	0.9386965	0.002047220	0.9335654	0.9450348	0.06133767
22	**D**	120	0.8737319	0.003002320	0.8663580	0.8827166	0.12630373
23	$\mathbf{DRM}_s$	480	0.8658798	0.008790841	0.8553493	0.8815749	0.13440782
24	$\mathbf{DR}^2\mathbf{M}_s$	1920	0.8638314	0.009177870	0.8497816	0.8813500	0.13647730
25	$\mathbf{DR}^3\mathbf{M}_s$	7680	0.8633066	0.009512230	0.8450996	0.8812937	0.13702374

TABLE S7: The statistics of the compactness of the cells in our proposed P_b. The compactness is heavily correlated to the compactness of the original faces of the input polyhdron. Thus, the icosahedron generally produces the most compact faces, followed by the pentakis dodecahedron. The disdyakis triacontahedron is a distant last, with the least compact faces of any of them.

			Compactness (d_C) of our proposed DGGS				
	P_b	# Cells	Mean	Std. Dev.	Min	Max	RMSE ↑
1	$\mathbf{I}$	20	0.9999999		0.9999999	0.9999999	0.00000006
2	$\mathbf{P}$	60	0.9589797		0.9589797	0.9589797	0.04102030
3	$\mathbf{IRM}_g$	80	0.9541176	0.004116452	0.9520594	0.9602923	0.04602066
4	$\mathbf{IRM}_s$	80	0.9541176	0.004116452	0.9520594	0.9602923	0.04602066
5	$\mathbf{IR}^2\mathbf{M}_g$	320	0.9447871	0.003636927	0.9375231	0.9475639	0.05532507
6	$\mathbf{PRM}_s$	240	0.9443524	0.002962599	0.9399114	0.9459854	0.05570672
7	$\mathbf{PRM}_g$	240	0.9437400	0.002439914	0.9400837	0.9450348	0.05629969
8	$\mathbf{IR}^2\mathbf{M}_s$	320	0.9434710	0.002901626	0.9381775	0.9465663	0.05659877
9	$\mathbf{I}(\mathbf{RM}_g)^2$	320	0.9426708	0.003025837	0.9382672	0.9475639	0.05740406
10	$\mathbf{IR}^3\mathbf{M}_g$	1280	0.9425553	0.002267740	0.9344409	0.9441464	0.05748872
11	$\mathbf{IR}^4\mathbf{M}_g$	5120	0.9420038	0.001749697	0.9337658	0.9432760	0.05802244
12	$\mathbf{PR}^2\mathbf{M}_s$	960	0.9406566	0.001881837	0.9351455	0.9427323	0.05937139
13	$\mathbf{IR}^3\mathbf{M}_s$	1280	0.9404844	0.001925206	0.9347128	0.9440226	0.05954629
14	$\mathbf{PR}^2\mathbf{M}_g$	960	0.9401846	0.001581977	0.9351111	0.9415958	0.05983503
15	$\mathbf{I}(\mathbf{RM}_g)^3$	1280	0.9398045	0.002735372	0.9347408	0.9441464	0.06025660
16	$\mathbf{PR}^3\mathbf{M}_s$	3840	0.9396622	0.001509109	0.9339556	0.9421862	0.06035641
17	$\mathbf{IR}^4\mathbf{M}_s$	5120	0.9396592	0.001544650	0.9338475	0.9432118	0.06036049
18	$\mathbf{PR}^4\mathbf{M}_s$	15360	0.9393913	0.001401808	0.9336583	0.9421495	0.06062482
19	$\mathbf{PR}^3\mathbf{M}_g$	3840	0.9393107	0.001278013	0.9339322	0.9410292	0.06070258
20	$\mathbf{PR}^4\mathbf{M}_g$	15360	0.9390931	0.001191588	0.9336503	0.9409994	0.06091850
21	$\mathbf{I}(\mathbf{RM}_g)^4$	5120	0.9390876	0.002661476	0.9338549	0.9432760	0.06097028
22	$\mathbf{D}$	120	0.8819678		0.8819678	0.8819678	0.11803217
23	$\mathbf{DRM}_s$	480	0.8678753	0.010148252	0.8594994	0.8815520	0.13241666
24	$\mathbf{DR}^2\mathbf{M}_s$	1920	0.8643113	0.009487952	0.8508555	0.8813500	0.13599931
25	$\mathbf{DR}^3\mathbf{M}_s$	7680	0.8634143	0.009587359	0.8453704	0.8812937	0.13691648



(a) $\mathbf{PM}_s$

(b) $\mathbf{PRM}_s$

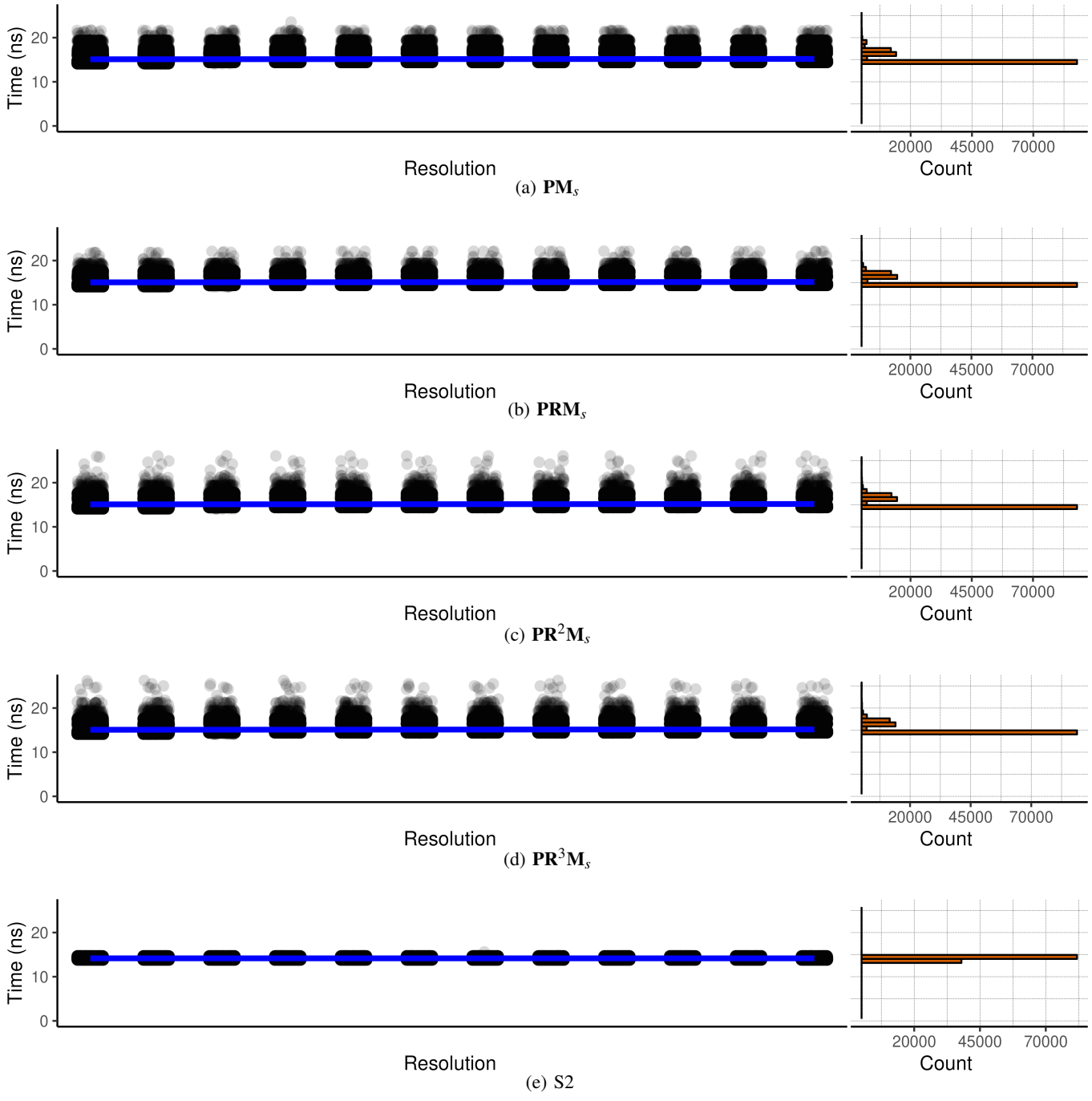
(c) $\mathbf{PR^2M}_s$

(d) $\mathbf{PR^3M}_s$

(e) S2

(f) H3 (note that y axis scale is larger than previous graphs)

Figure S2: Performance of the PointToCell operation (with conversion from WGS84) for our MBD with the refined pentakis dodecahedron (a)-(d), S2 (e) and H3 (f). Note that the scale of the y axis for (f) is different due to its significant difference in performance. Also note that at all levels of resolution and for all proposed base polyhedra, the mean time of the operation is constant. Benchmarked across 10,000 points per resolution. Units in nanoseconds.



(a) $\mathbf{PM}_s$

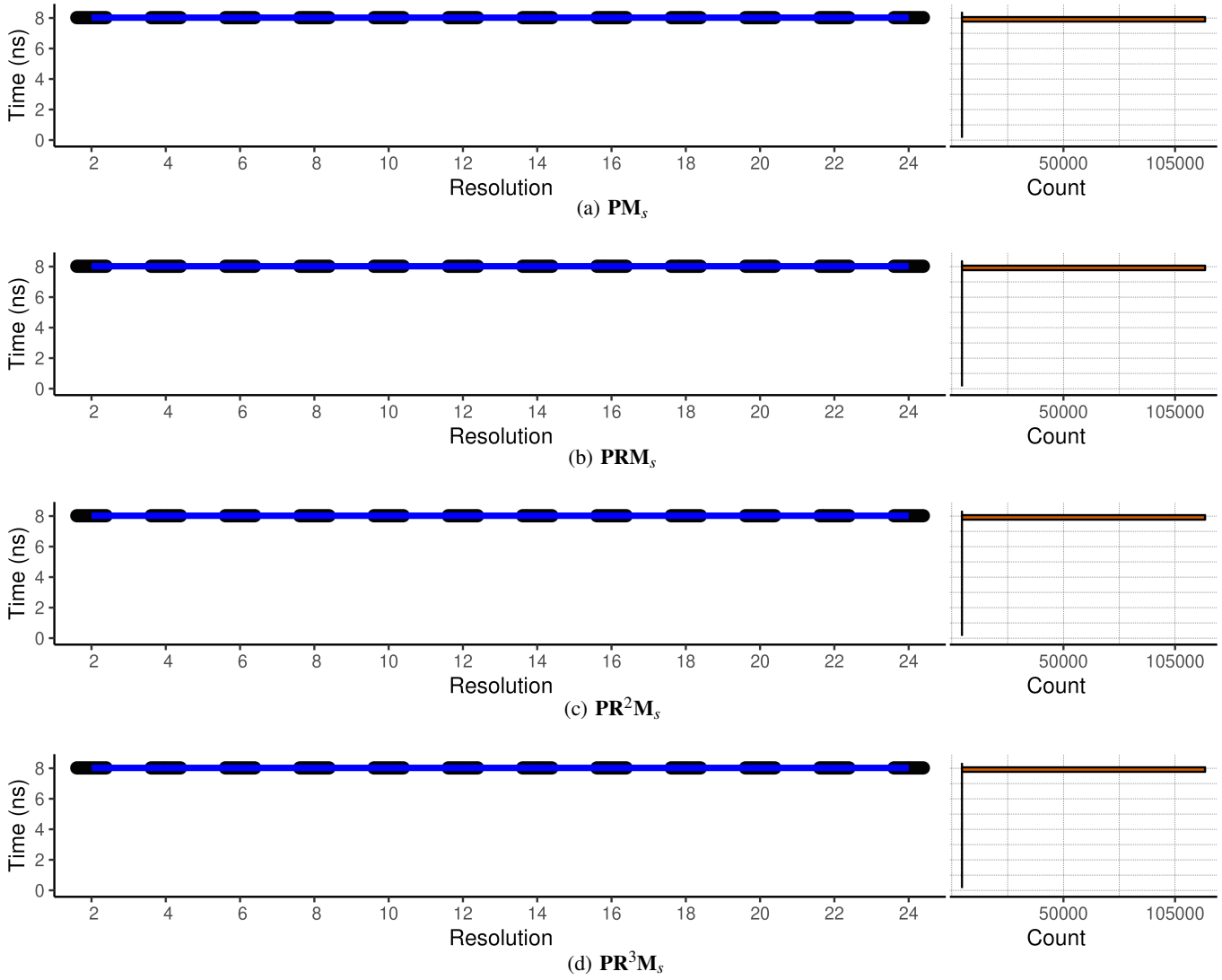
(b) $\mathbf{PRM}_s$

(c) $\mathbf{PR}^2\mathbf{M}_s$

(d) $\mathbf{PR}^3\mathbf{M}_s$

(e) S2

Figure S3: Performance of the POINTTOCELL operation (without conversion from WGS84) for our MBD with the refined pentakis dodecahedron (a)-(d) and S2 (e). Note that at all levels of resolution and for all proposed base polyhedra, the mean time of the operation is constant. Benchmarked across 10,000 points per resolution. Units in nanoseconds.



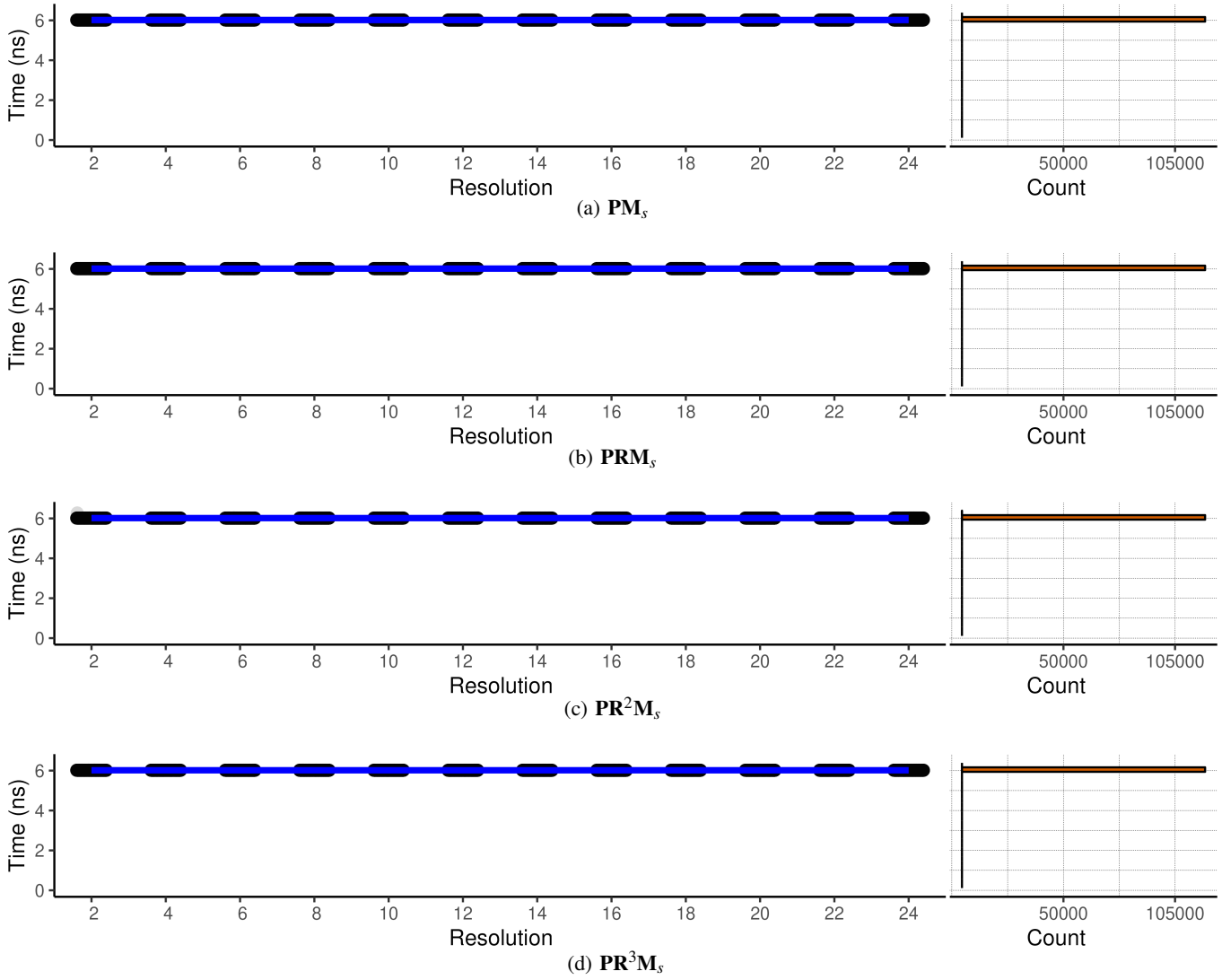
(a) $\mathbf{PM}_s$

(b) $\mathbf{PRM}_s$

(c) $\mathbf{PR^2M}_s$

(d) $\mathbf{PR^3M}_s$

Figure S4: Performance of the CHILDREN operation for our MBD with the refined pentakis dodecahedron (a)-(d). Benchmarked across 10,000 points per resolution. Units in nanoseconds.



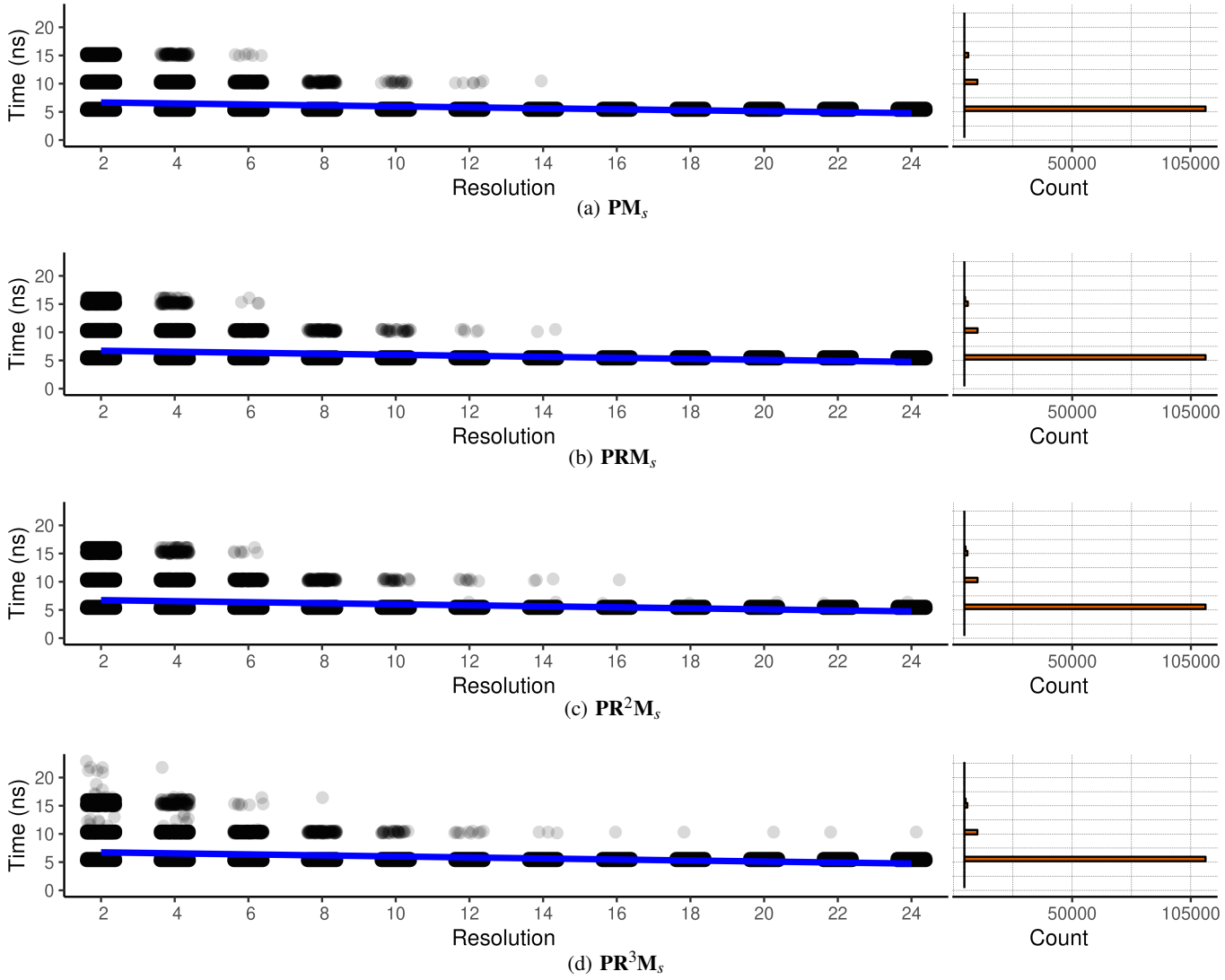
(a) $\mathbf{PM}_s$

(b) $\mathbf{PRM}_s$

(c) $\mathbf{PR}^2\mathbf{M}_s$

(d) $\mathbf{PR}^3\mathbf{M}_s$

Figure S5: Performance of the PARENT operation for our MBD with the refined pentakis dodecahedron (a)-(d). Benchmarked across 10,000 points per resolution. Units in nanoseconds.



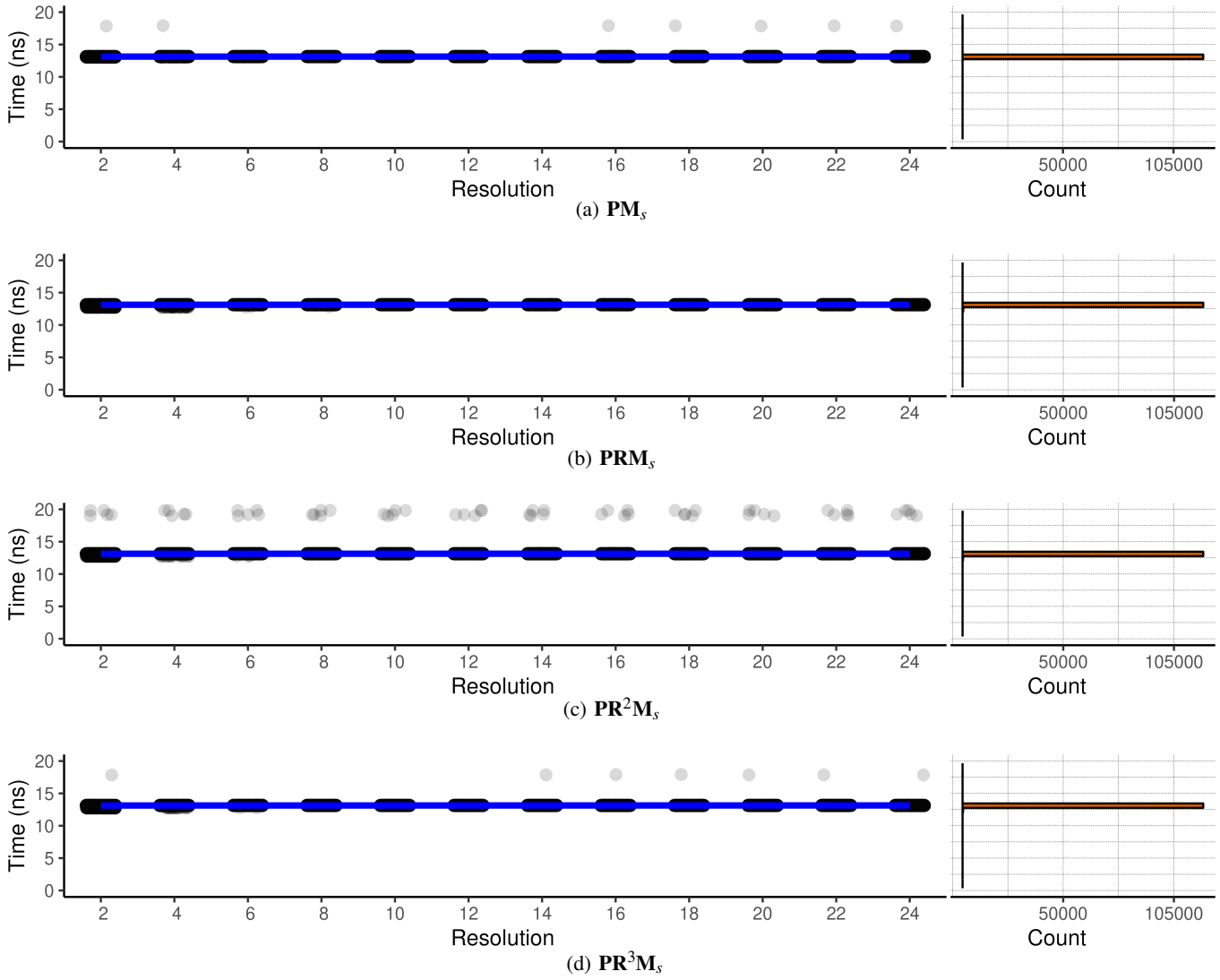
(a) $\mathbf{PM}_s$

(b) $\mathbf{PRM}_s$

(c) $\mathbf{PR^2M}_s$

(d) $\mathbf{PR^3M}_s$

Figure S6: Performance of the NEIGHBOR operation for our MBD with the refined pentakis dodecahedron (a)-(d). Benchmarked across 10,000 points per resolution. Units in nanoseconds.



(a) $\mathbf{PM}_s$

(b) $\mathbf{PRM}_s$

(c) $\mathbf{PR^2M}_s$

(d) $\mathbf{PR^3M}_s$

Figure S7: Performance of the SPHERICALGEOMETRYFROMINDEX operation for our MBD with the refined pentakis dodecahedron (a)-(d). Benchmarked across 10,000 points per resolution. Units in nanoseconds.