Culling for Extreme-Scale Segmentation Volumes: A Hybrid Deterministic and Probabilistic Approach
SUPPLEMENTARY MATERIAL

Johanna Beyer, Haneen Mohammed, Marco Agus, Ali K. Al-Awami, Hanspeter Pfister, and Markus Hadwiger

10 EVALUATION RESULTS

In this supplementary material, we report more detailed evaluation results for our hybrid culling method. In Sec. 10.1, we first evaluate culling queries for volume rendering in more detail, by analyzing the impact of different false positive settings in Bloom filters on performance and memory footprint. We also list culling results for a brute-force, non-hierarchical algorithm for comparison. In Sec. 10.2, we analyze spatial queries, and list the memory footprint of our hybrid approach in comparison to uncompressed bit strings. In Sec. 10.3, we analyze the memory impact of hierarchical query pruning.

Furthermore, we give more details on probabilistic culling with Bloom filters in Sec. 10.4, and give detailed memory footprint information of our label list tree data structure in Sec. 10.5. Finally, we also list data set statistics and give details on the distribution of labels in the different resolution levels of the evaluated volumes in Sec. 10.6.

10.1 Culling Queries for Rendering

Table 3 shows the culling performance of our system when using Bloom filters with different false positive rates, and compares it to a brute-force, non-hierarchical approach. The brute-force approach iterates linearly over all blocks of the current resolution level (the one currently selected for rendering), and culls all blocks against the query.

We compare false positive (FP) rates of 5%, 10%, and 25%, respectively. Overall, the number of nodes that are touched during culling increases when the false positive rate is increased. However, our hybrid method is consistently faster and more memory-efficient than a non-hierarchical brute-force approach.

10.2 Spatial Queries

Table 4 shows detailed results of our culling method for performing spatial proximity queries. We evaluate four different queries with different cardinalities and selected label IDs (Q1-Q4). We use a block size of 512^3, which allows us to efficiently handle very large data sets. We report the size of the label lists that are touched during culling and compare uncompressed bit strings (column 5) with our hybrid label list format (column 6).

Overall, our method consistently has a smaller memory footprint for the accessed label lists than uncompressed bit strings. The number of nodes touched is the same for both approaches, as spatial queries are always performed in a hierarchical way, even when using uncompressed bit strings. The SEM Mouse Cortex data set is relatively sparse (i.e., only contains 4,125 labels), therefore, the overall size of examined label lists is very small. The Mouse Cortex 2 and the Phantom Spheres 2 data sets are very dense, resulting in larger sizes for the label lists.

10.3 Query Pruning

Fig. 11 shows the effect of query pruning on the size of the touched label lists during culling. We evaluate different queries, with cardinalities of 2, 10, 100, and 1,000 for the Mouse cortex 2 data set. Green lines depict queries that are not getting pruned, while blue lines depict

Table 3. ***Culling performance for volume rendering with empty space skipping with different Bloom filter false positive rates.*** We compare our empty space skipping approach with hybrid label lists to a non-hierarchical culling approach using bit strings. We list the number of nodes touched for culling, the size of the touched label lists, and the evaluation time for two different queries (Q1: 2 labels, Q2: 1,000 labels). False positive (FP) rates: 5%, 10%, and 25%, $c_B = 10$.

data set	culling method		# nodes touched	label data touched	time (ms)
KESM Mouse Brain	hybrid	Q1	137 (0.76 %)	217 KB	3.4
	FP 5%	Q2	3,333 (18.7 %)	705 KB	26.8
	hybrid	Q1	261 (1.6 %)	203 KB	7.6
	FP 10%	Q2	3,109 (19.5 %)	647 KB	25
	hybrid	Q1	373 (3.0 %)	127 KB	6.6
	FP 25%	Q2	3,797 (31.0 %)	605 KB	28.2
	non-	Q1	13,895 (86.9 %)	381 MB	153.2
	hierar.	Q2	13,895 (86.9 %)	381 MB	142.6
SEM Mouse Cortex	hybrid	Q1	141 (1.5 %)	54 KB	5
	FP 5%	Q2	3,269 (34.3 %)	257 KB	28.4
	hybrid	Q1	141 (1.5 %)	48 KB	7
	FP 10%	Q2	3,269 (34.3 %)	188 KB	27.2
	hybrid	Q1	285 (3.0 %)	56 KB	9.2
	FP 25%	Q2	3,269 (34.3 %)	188 KB	24.2
	non-	Q1	8,190 (85.9 %)	4.1 MB	7.4
	hierar.	Q2	8,190 (85.9 %)	4.1 MB	6.6
Mouse Cort. 2	hybrid	Q1	449 (4.7%))	6.0 MB	9.2
	FP 5%	Q2	1,889 (19.6 %)	12.1 MB	15.6
	hybrid	Q1	417 (1.9 %))	13.4 MB	12.8
	FP 10%	Q2	1,713 (7.83 %)	27.7 MB	25.2
	hybrid	Q1	865 (9.8%))	5.0 MB	11.0
	FP 25%	Q2	1,737 (19.6 %)	11.5 MB	16.6
	non-	Q1	12,986 (87.5 %)	25.4 GB	7,019
	hierar.	Q2	12,986 (87.5 %)	25.4 GB	6,549
Phantom Spheres	hybrid	Q1	369 (3.1%)	3.0 MB	7
	FP 5%	Q2	9,657 (81.9%)	18.9 MB	970.0
	hybrid	Q1	273 (1.74%)	3.8 MB	8
	FP 10%	Q2	12,329 (78.7%)	25.7 MB	92.4
	hybrid	Q1	737 (3.74%)	4.0 MB	16.2
	FP 25%	Q2	15,233 (77.3%)	27.9 MB	111.0
	non-	Q1	16,332 (87.5%)	9.8 GB	3,357
	hierar.	Q2	16,332 (87.5%)	9.8 GB	2,976

- *Johanna Beyer and Hanspeter Pfister are with Harvard University, Cambridge, MA, USA. E-mail: {jbeyer, pfister}@seas.harvard.edu*
- *Marco Agus, and Markus Hadwiger are with King Abdullah University of Science and Technology (KAUST), Thuwal, 23955-6900, Saudi Arabia. E-mail: {markus.hadwiger, marco.agus}@kaust.edu.sa.*
- *Haneen Mohammed is with Harvard University and KAUST. E-mail: haneen.mohammed@kaust.edu.sa*
- *Ali K. Al-Awami is with KAUST and Saudi Aramco, Dhahran, Saudi Arabia. E-mail: awami.ali@gmail.com*

queries that are getting pruned during hierarchical traversal (Fig. 11).

Query pruning significantly reduces the memory footprint of the used label lists during culling. The reason for this is that pruning continuously minimizes the size of the query. Subsequently, smaller queries result in our query-adaptive culling method requesting Bloom filter-based label lists instead of Roaring (based on parameter c_B). Since we only store label lists as Bloom filters when they have a lower memory footprint than Roaring bitmaps, this explains the overall smaller memory size.

Table 4. ***Performance of spatial queries.*** We compare the evaluation of four different spatial queries. To perform a spatial query, we find all nodes containing a certain label or set of labels. For example, all nodes for computing the minimum distance between a set of objects. We compare our label lists to standard uncompressed bit strings. We list the overall number of nodes in the label list tree, the number of nodes touched during culling, the overall size of the required bit strings, and the size of our label lists. We compare four different queries, Q1 has a cardinality of 1, Q2 a cardinality of 2, Q3 contains 50 spatially close labels, and Q4 contains 50 randomly selected labels. Block size is 512^3.

Data set	Query	#nodes overall	#nodes touched	Bitstring size (KB)	Label list size (KB)
KESM	Q1	434	45	1,230	284
	Q2	434	53	1,449	316
	Q3	434	53	1,449	273
	Q4	434	117	3,199	421
SEM Mouse Cortex	Q1	11,525	37	17	16
	Q2	11,525	57	29	18
	Q3	11,525	313	158	42
	Q4	11,525	993	500	77
Mouse Cortex 2	Q1	589	45	92,120	17,405
	Q2	589	45	92,120	17,405
	Q1	589	109	223,135	28,904
	Q2	589	253	517,921	43,191
Phantom Spheres 2	Q1	77	21	12,579	4,306
	Q2	77	29	17,371	5,490
	Q1	77	45	26,955	7,908
	Q2	77	77	46,123	12,696

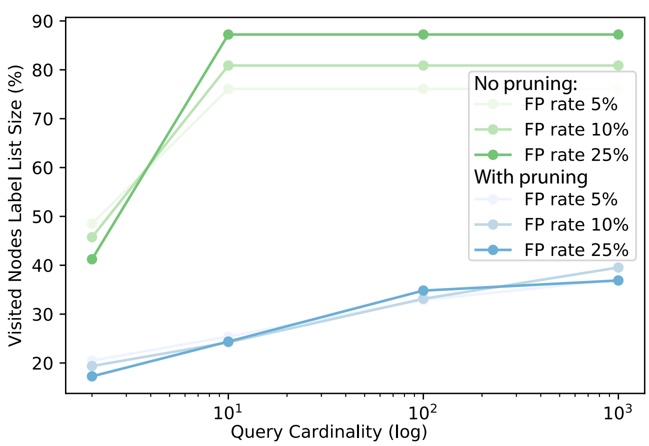


Fig. 11. ***Memory footprint of query pruning.*** We compare the memory footprint (i.e., the size of all nodes touched during culling) with and without query pruning for different queries. Query cardinality ranges from 2 to 1,000 labels (x-axis), and we report the size of the touched nodes as a percentage of the total label list tree size. Query evaluation with pruning (blue lines) consistently results in a smaller memory footprint. Query evaluation without pruning (green lines) results in a larger memory footprint. This is due to the fact that smaller queries tend to request label lists as Bloom filters which are typically smaller in size than Roaring label lists. Data set: Mouse Cortex 2.

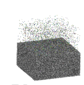
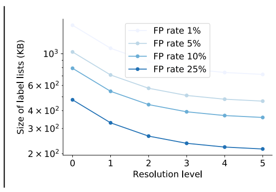
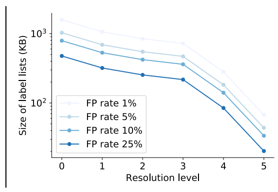
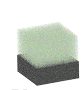
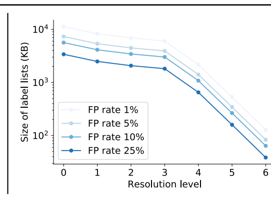
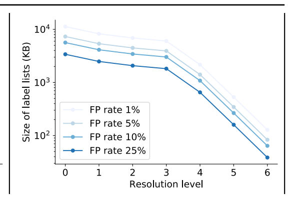
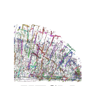
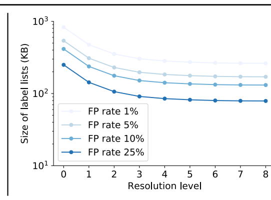
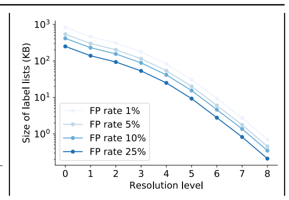
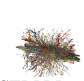
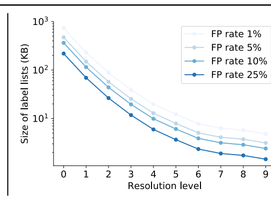
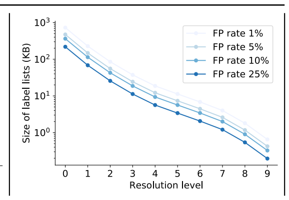
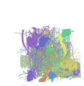
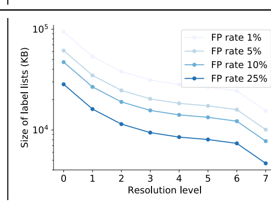
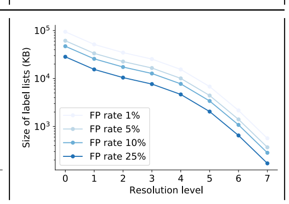
10.4 Bloom Filters

Table 5 gives detailed statistics about Bloom filters. For each data set, we evaluate Bloom filter sizes for different resolution levels (the full resolution is level 0), for different false positive rates. A high false positive rate leads to smaller label lists. However, it also causes more nodes to be touched during hierarchical culling (as nodes might not get culled due to false positives). Column two in Table 5 analyzes resolution-independent label lists, and column three analyzes resolution-adjusted label lists. Note that the size of resolution-adjusted label lists decreases significantly for lower-resolution levels.

10.5 Label list tree: Memory footprint and data representations

Table 6 lists the memory footprint of the label list tree when using different data representations for the label lists. We compare the memory footprint of label lists that are solely represented as uncompressed bit strings, Roaring bitmaps, Roaring bitmaps with delta encoding, and our hybrid approach with different false positive settings in the line charts in columns two and four. We analyze the exact memory consumption of the label list tree for every resolution level, and compare

Table 5. ***Bloom filter statistics.*** We show statistics for Bloom filters for representing resolution-independent (column 2) and resolution-adjusted (column 3) label lists, respectively. The graphs show the Bloom filter memory size in KB for each resolution level and for different false positive rates, ranging from 1% to 25%. Block size is 32^3.

Data set	Bloom filter size per level Res-indep (KB)	Bloom filter size per level Res-adjust (KB)
 Phantom spheres 1k		
 Phantom spheres 2k		
 KESM Mouse Brain		
 SEM Mouse Cortex		
 Mouse Cortex 2		

the sizes of resolution-independent (column 2) and resolution-adjusted label lists (column 4), respectively. Note that even for uncompressed bit strings we only store the bits inside the min/max range of label IDs, as computed for each individual block. For example, if a 16-bit label volume only contains IDs from 0-4095, uncompressed bit strings will be stored with length 2^{12} instead of 2^{16}. Our hybrid approach consistently uses less memory than regular bit strings for both resolution-independent and resolution-adjusted label lists.

In columns 3 and 5, we evaluate our hybrid data representation in more detail. The stacked bar chart depicts the distribution of the number of nodes encoded with Roaring, delta, and Bloom filters in our hybrid approach. For each resolution level we show 3 bars, representing our hybrid approach with different false positive settings (5%, 10%, and 25%). Within each bar, we depict the distribution of nodes that are represented with Roaring, delta, and Bloom filters. Bloom filters are primarily used for nodes with a limited number of labels (i.e., low label list cardinality). For resolution-independent label lists this limits the use of Bloom filters to the higher resolution levels. In resolution-adjusted label lists, the number of labels is always bounded by the number of voxels in the corresponding volume block (e.g., 32^3), and, therefore, Bloom filters might also be used in lower resolution levels.

In Fig. 12, we analyze the memory footprint between deterministic and probabilistic culling (both using our hierarchical culling method). We depict the difference in the number of visited nodes (solid lines, left-hand y-axes) as well as the memory footprint of the visited nodes (dashed lines, right-hand y-axes). Probabilistic culling uses our hybrid method, while for deterministic culling we have disabled the use of Bloom filters. Probabilistic culling consistently needs less memory, even though it visits more nodes compared to pure deterministic culling. This is exactly what we would expect: Bloom filters are very compact and fast to evaluate, however, their false positive rate causes more nodes to be touched during hierarchical tree traversal.

10.6 Data Set and Label Distribution Statistics

Table 7 and Table 8 give details on the distribution (occurrence) of labeled segments in the different resolution levels of our data sets.

Table 7 lists general information for each data set (column 2), the overall size of label lists in different data representations (column 3), and details of the distribution of labels within volume blocks (column 4 and 5). We evaluate resolution-independent (column 4) and resolution adjusted label lists (column 5), and depict the average, min, and max number of labels per block with box plots.

Note that volumes with many small structures, such as the Phantom Spheres or the Mouse Cortex 2 data sets, exhibit less variance in the label count per block over all resolution levels. The number of labels per node in the SEM Mouse Cortex data set, on the other hand, is more spread out. This can be explained by the fact that some regions in the volume have been densely labeled, whereas other regions barely contain any labels. The Mouse Cortex 2 data set has been labeled by an automatic dense labeling approach favoring many small segments. Therefore, the number of labels per node is more consistent within each resolution level.

Table 8 gives detailed numbers on the label distribution in blocks. We list again the min, max, average, and standard deviation of the number of labels per block, for each resolution level and for resolution-independent as well as resolution-adjusted label lists.

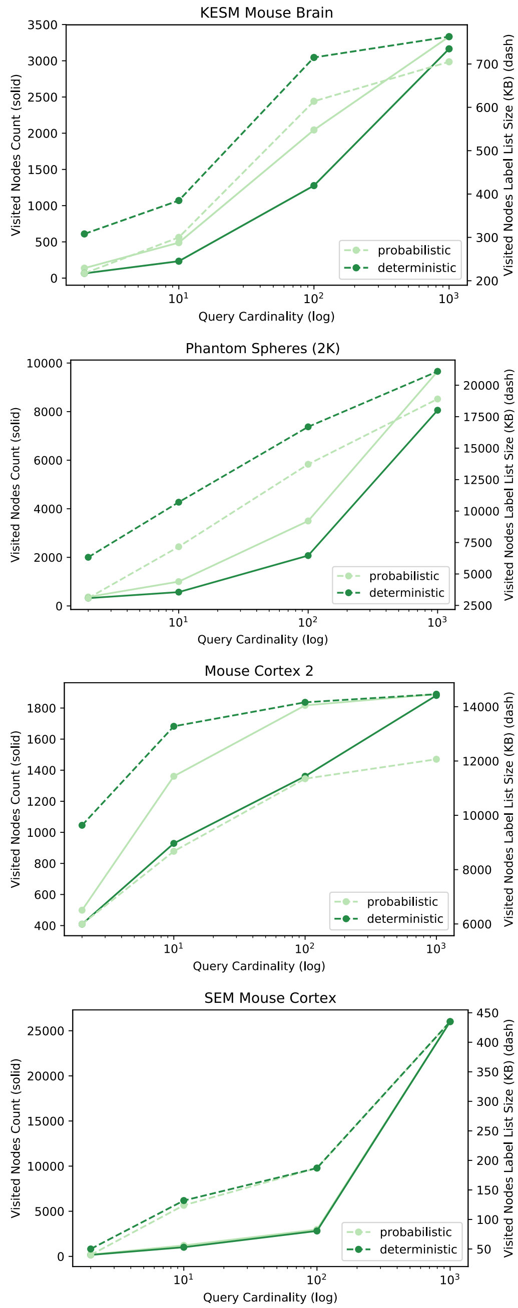


Fig. 12. ***Deterministic vs. probabilistic label list memory footprints during culling.*** We analyze the difference in the number of visited nodes (solid lines, left-hand y-axes) as well as the memory footprint of the visited nodes (dashed lines, right-hand y-axes) for our hybrid culling method using probabilistic as well as deterministic culling. Probabilistic culling (light green lines) consistently needs less memory, even though it visits more nodes compared to pure deterministic culling.

Table 6. ***Memory footprint of label lists and data encoding in label list trees.*** Columns 2 and 4 show the memory footprint (in KB) of label lists for various encoding strategies per resolution level. We compare standard bit strings, Roaring bitmaps, Roaring bitmaps with delta encoding, and our hybrid representation (with different FP rates). Column 2: Resolution-independent label lists. Column 4: Resolution-adjusted label lists. In columns 3 and 5 we analyze the distribution of the different data representations in our hybrid strategy in a stacked bar chart. In each bar we show the number of nodes that are encoded with Roaring bitmaps, Roaring bitmaps with delta encoding, and Bloom filters, respectively. We show results for three different FP settings and per resolution level (i.e., three bars per resolution level). The bars correspond to FP settings of 5%, 10%, and 25%; from left to right). Column 3: Resolution-independent label lists. Column 5: Resolution-adjusted label lists. Block size is 32^3.

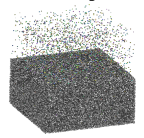
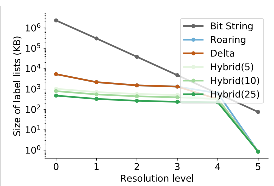
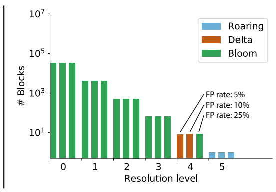
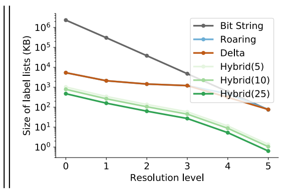
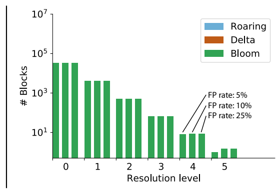
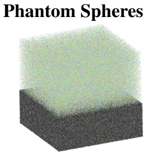
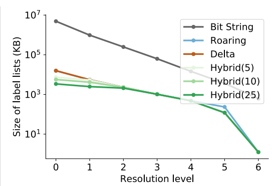
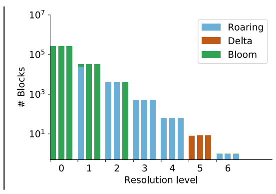
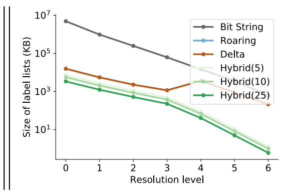
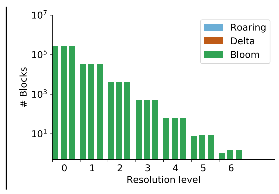
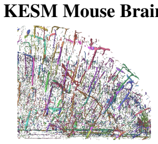
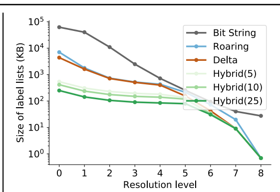
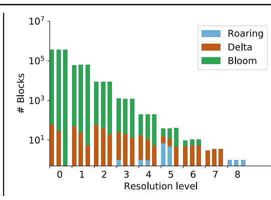
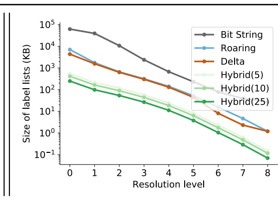
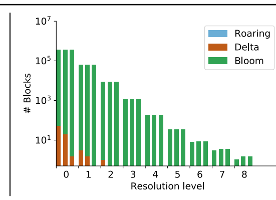
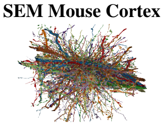
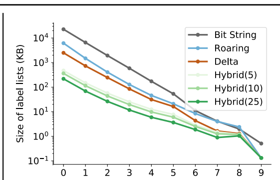
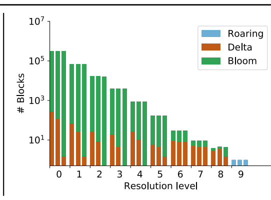
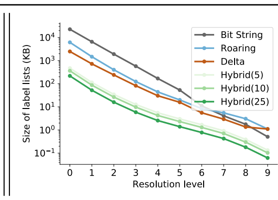
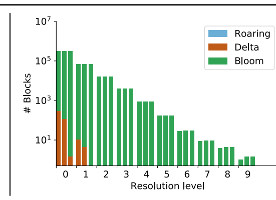
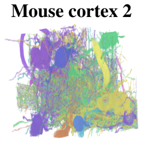
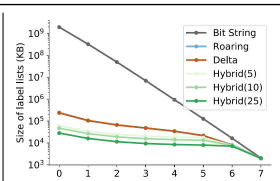
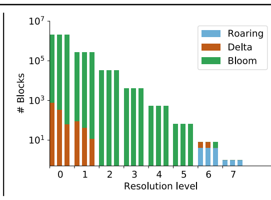
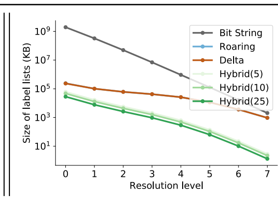
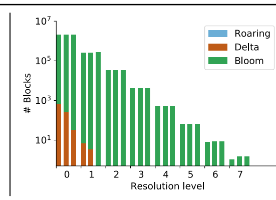
Data set	Label size per level Res-indep	Distribution per level Res-indep	Label size per level Res-adjust	Distribution per level Res-adjust
Phantom Spheres 				
Phantom Spheres 2 				
KESM Mouse Brain 				
SEM Mouse Cortex 				
Mouse cortex 2 				

Table 7. ***Data set details and label distribution statistics per data set and resolution level.*** Col. 2 gives general data set information (resolution, size, number of resolution levels). Col. 3 lists the total sizes of the label list tree for different encoding strategies: uncompressed bit strings, Roaring bitmaps, Roaring bitmaps with delta encoding, and our hybrid strategy (with different false positive rates for Bloom filters). In cols. 4 and 5 we show box plots of the distribution of the number of labels per node, for resolution-independent (col. 4) and resolution-adjusted (col. 5) label lists, respectively. Each box plot shows per resolution level the min, max, and average number of labels per block. Block size is 32^3.

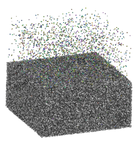
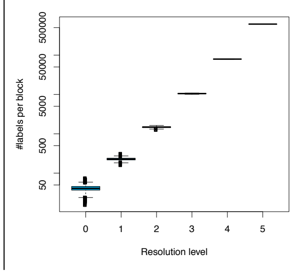
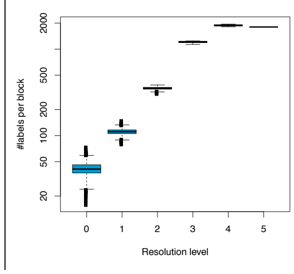
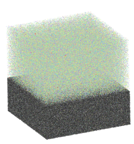
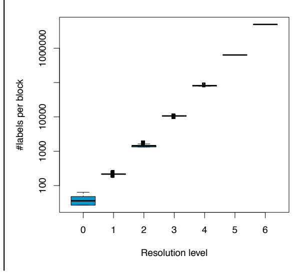
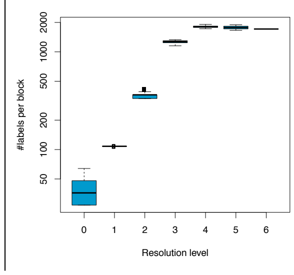
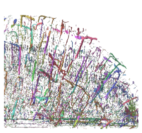
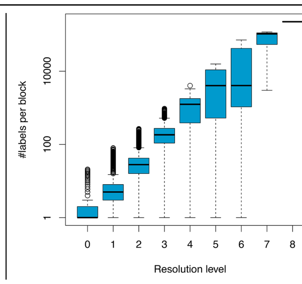
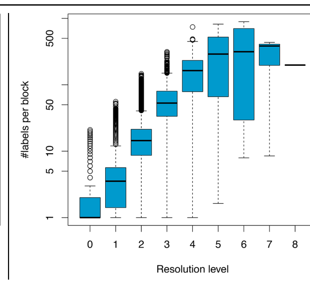
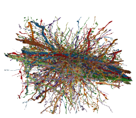
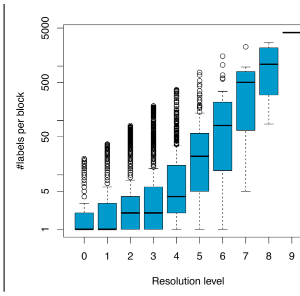
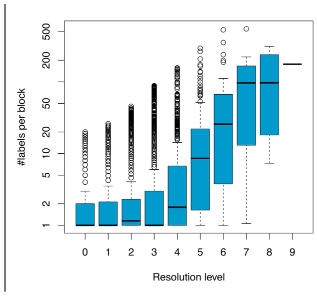
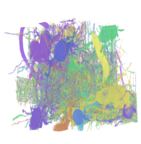
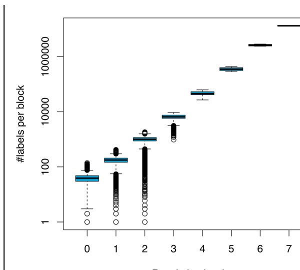
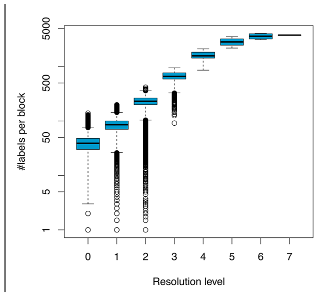
Data set	Description	Label list total sizes (MB)	Label count per level Res-indep	Label count per level Res-adjust
	**Phantom Spheres** 1,024 x 1,024 x 1,024 Images: 1 GB (8 bit) Labels: 3 GB (24 bit) Resolution levels: 6 # Labels: 614 K	**Res-indep / Res-adjust** Bitstring: 2622.65 / 2622.56 Roaring: 10.77 / 10.48 Delta Roaring: 10.47 / 10.32 Hybrid (5% FPR): 3.07 / 2.90 Hybrid (10% FPR): 2.43 / 2.22 Hybrid (25% FPR): 1.49 / 1.34		
	**Phantom Spheres 2** 2,048 x 2,048 x 2,048 Images: 8 GB (8 bit) Labels: 24 GB (24 bit) Resolution levels: 7 # Labels: 4.91 M	**Res-indep / Res-adjust** Bitstring: 6056.06 / 6055.96 Roaring: 24.7 / 28.78 Delta Roaring: 24.6 / 28.78 Hybrid (5% FPR): 16.2 / 17.5 Hybrid (10% FPR): 13.4 / 14.3 Hybrid (25% FPR): 9.4 / 9.7		
	**KESM Mouse Brain** 2,380 x 9,216 x 2,039 Images: 42.7 GB (8 bit) Labels: 128.1 GB (24 bit) Resolution levels: 9 # Labels: 224 K	**Res-indep / Res-adjust** Bitstring: 114.4 / 113.1 Roaring: 10.6 / 9.77 Delta Roaring: 7.6 / 6.9 Hybrid (5% FPR): 1.6 / 1.21 Hybrid (10% FPR): 1.25 / 0.92 Hybrid (25% FPR): 0.78 / 0.56		
	**SEM Mouse Cortex** 21,494 x 25,790 x 1,850 Images: 955 GB (8 bit) Labels: 489 GB (16 bit) Resolution levels: 10 (11 levels for images) # Labels: 4,107	**Res-indep / Res-adjust** Bitstring: 31.3 / 31.2 Roaring: 8.1 / 8.06 Delta Roaring: 3.53 / 3.51 Hybrid (5% FPR): 0.71 / 0.71 Hybrid (10% FPR): 0.55 / 0.55 Hybrid (25% FPR): 0.33 / 0.33		
	**Mouse Cortex 2** 4,096 x 4,096 x 4,096 Images: 64 GB (8 bit) Labels: 192 GB (24 bit) Resolution levels: 8 # Labels: 13.25 M	**Res-indep / Res-adjust** Bitstring: 2.27TB / 2.27TB Roaring: 515.4 / 477.2 Delta Roaring: 509.8 / 471.2 Hybrid (5% FPR): 182.2 / 146.7 Hybrid (10% FPR): 142.2 / 112.7 Hybrid (25% FPR): 88.8 / 67.9		

Table 8. **Data set label distribution statistics.** We give information about the minimum, maximum, and average number of labels per block. We list the statistics per resolution level, and in each level give statistics for resolution-independent and resolution-adjusted label lists, respectively. Resolution level L0 corresponds to the highest resolution of each data set.

	Phantom Spheres 2				**KESM Mouse Brain**				**SEM Mouse Cortex**				**Mouse Cortex 2**			
L0	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind	27	64	36.9	9.1	1	21	1.98	1.8	1	20	1.96	2	1	140	38.5	14.9
res-adj	27	64	36.9	9.1	1	21	1.98	1.8	1	20	1.96	2	1	140	38.5	14.9
L1	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind	200	237	215.8	1.98	1	80	6.6	6.9	1	37	2.8	3.9	1	422	174.8	53.3
res-adj	107	108	108	0.01	1	56	4.6	4.7	1	26	2.1	2.7	1	199	84	25.5
L2	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind	1.3K	1.7K	1.4K	101.8	1	268	35.2	32	1	81	4.5	8.4	1	1.8K	992	237
res-adj	332	413	360	22.5	1	147	18	16	1	45	2.7	4.7	1	418	227	54
L3	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind	10.2K	10.7K	10.6K	91.3	1	953	209	155	1	186	8.3	19.7	977	9.5K	6.5K	1.3K
res-adj	1.1K	1.2K	1.2K	44	1	314	62.5	48	1	88	4.2	9.5	92	950	665	128
L4	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind	77.8K	85.2K	80.7K	1.5K	1	4K	1.2K	885.4	1	370	19.4	46.3	26.9K	62.5K	47.2K	7.2K
res-adj	1.7K	1.9K	1.8K	46	1	743	174	123	1	157	8.4	19.7	860	2.1K	1.6K	253
L5	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind	632.7K	636.1K	634.6K	1.6K	1	15.7K	5.8K	5.5K	1	765	61.2	114.5	288K	438.6K	357K	46.8K
res-adj	1.6K	1.9K	1.8K	85	2	818	316	265	1	294	23.4	44.1	2.2K	3.5K	2.8K	397
L6	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind	4.9M	4.9M	4.9M	0	1	71.2K	22.8K	28.4K	1	1526	230	355	2.4M	2.88M	2.61M	0.2M
res-adj	1.7K	1.7K	1.7K	0	8	890	374	380	1	531	76.5	122	3.1K	4.1K	3.6K	431
L7					*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
res-ind					3K	117.9K	75.1K	62.8K	5	2.3K	600	722	13.2M	13.2M	13.2M	0
res-adj					9	434	276	233	1	548	135	175	3.8K	3.8K	3.8K	0
L8					*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*				
res-ind					224K	224K	224K	0	86	2.7K	1.2K	1.2K				
res-adj					198	198	198	0	7	312	128	141				
L9									*min*	*max*	*avg*	*std*				
res-ind									4125	4125	4125	0				
res-adj									176	176	176	0				