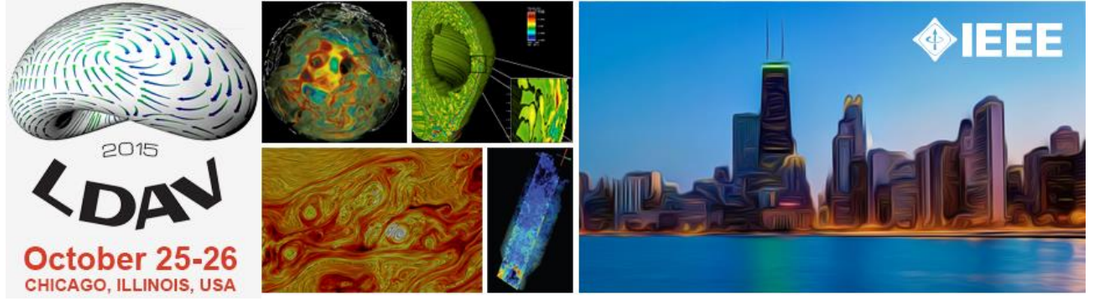
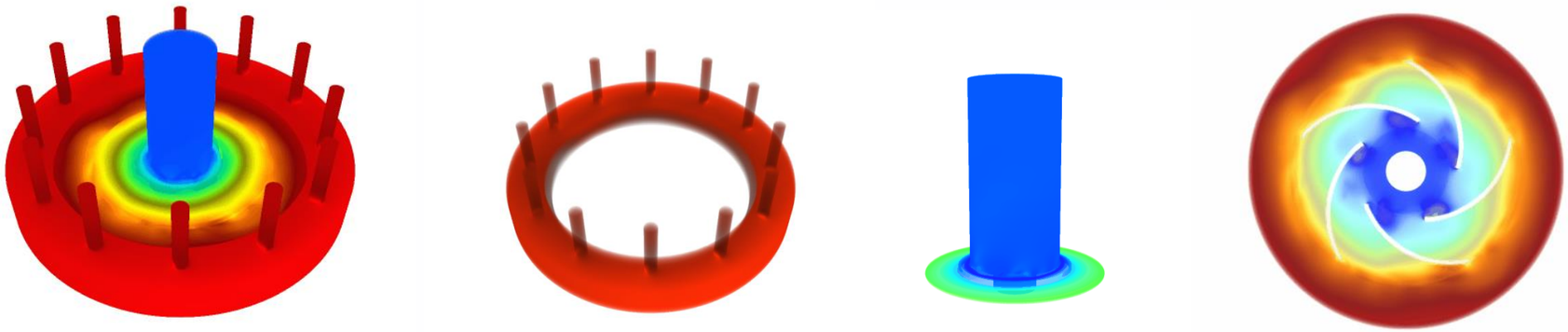


Cylindrical Acceleration Structures for Large Hexahedral Volume Visualization

Junpeng Wang[1,2] Mai Elshehaly[2] Yong Cao[2]

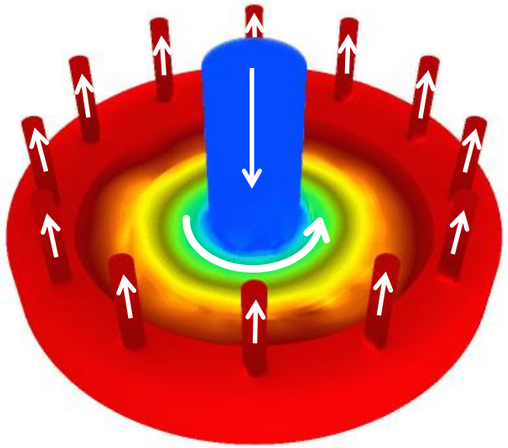
[1] The Ohio State University [2] Virginia Tech



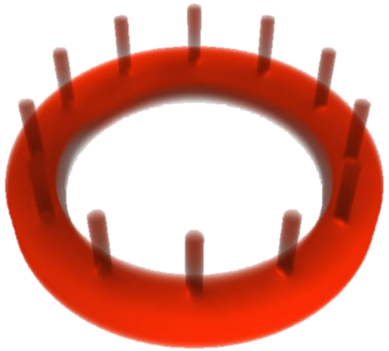
THE OHIO STATE UNIVERSITY

VirginiaTech
Invent the Future

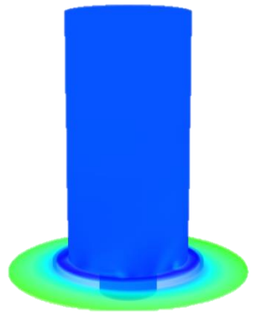
Problem



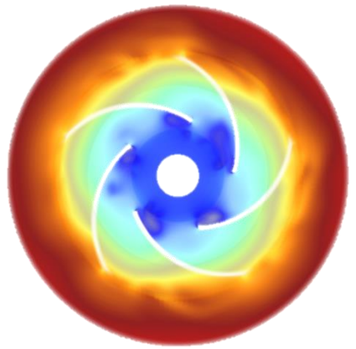
Large
Irregular
Unevenly Distributed



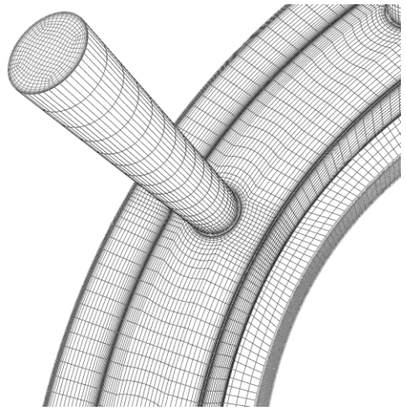
diffuser

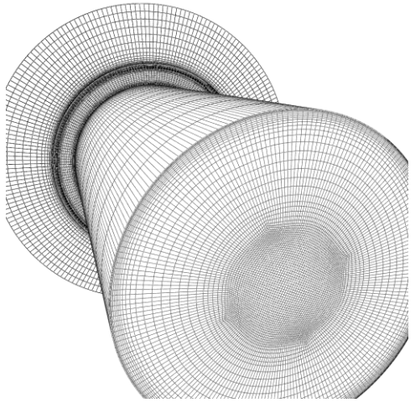


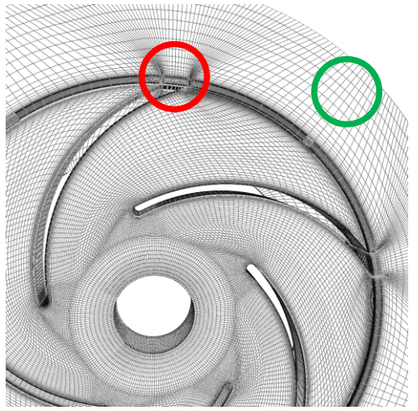
inlet



rotor

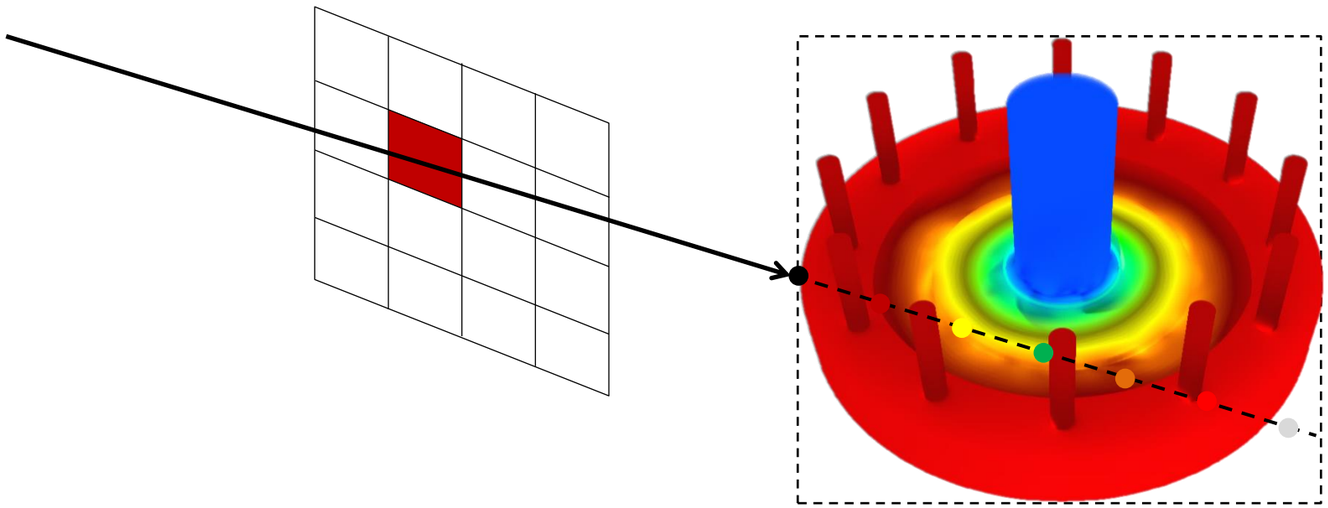


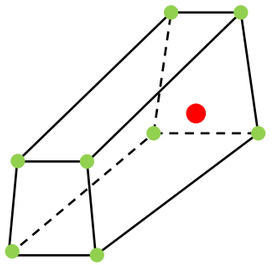




Number of Vertices:	3.6 million	0.9 million	2.2 million
Number of Hexahedra:	3.4 million	0.9 million	2.1 million

Problem

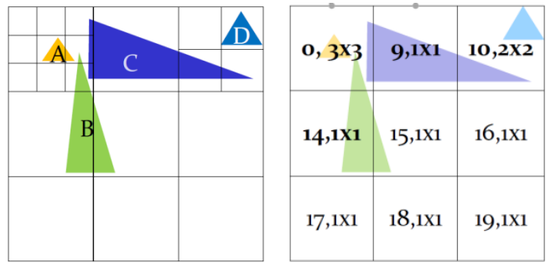




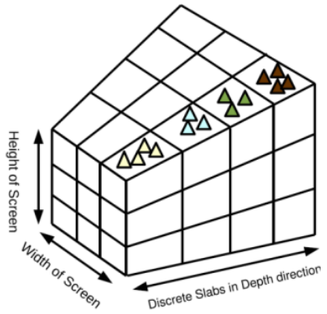
Related Work

FLAT GRIDS

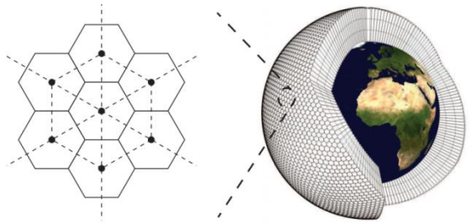
Two-level Grid
[Kalojanov et al. 2011]



Perspective Grid
[Guntury and Narayanan 2012]



Spherical Geodesic Grids
[Xie et al. 2012]



HIERARCHICAL TREES

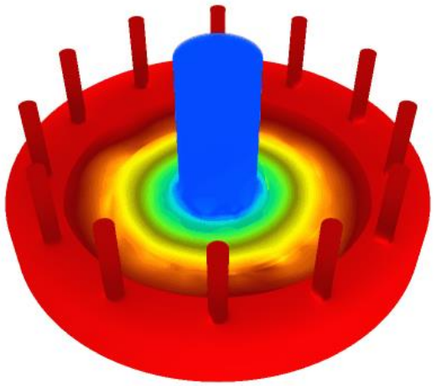
***k*D-tree**
[Hunt et al. 2006]
[Zhou 2011]

Octree
[Karras 2012]
[Zhou et al. 2011]

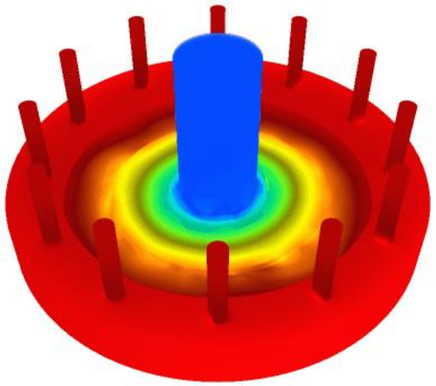
Bounding Volume Hierarchy (BVH)
[Lauterbach et al. 2009]
[Garanzha et al. 2011]
[Kopta et al. 2012]
[Karras and Aila 2013]

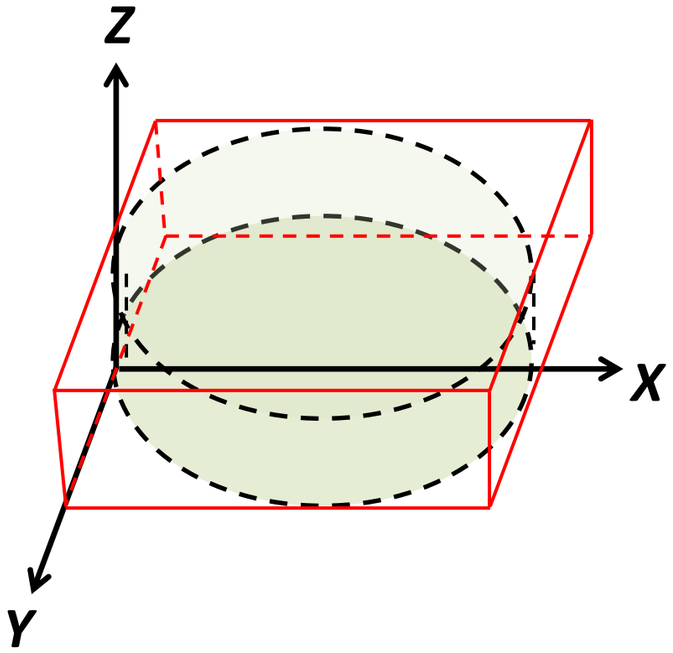
Surface Area Heuristic (SAH)
[Goldsmith and Salmon 1987]
[MacDonald and Booth 1990]
VVH [Wald et al. 2004]

Motivation

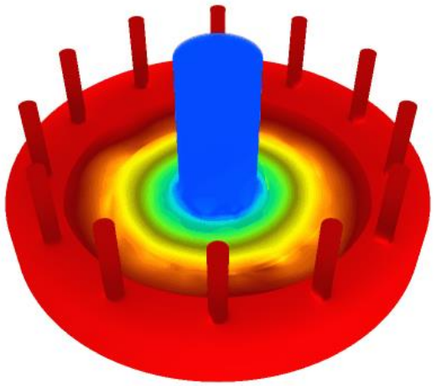


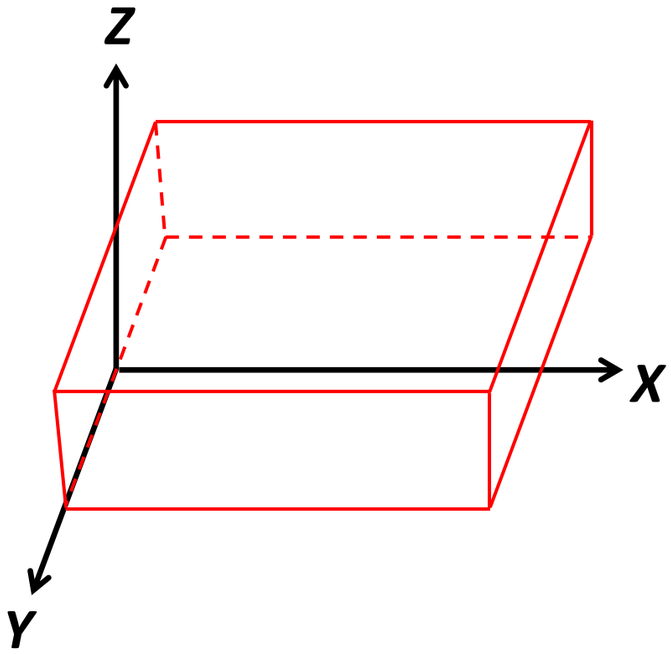
Motivation

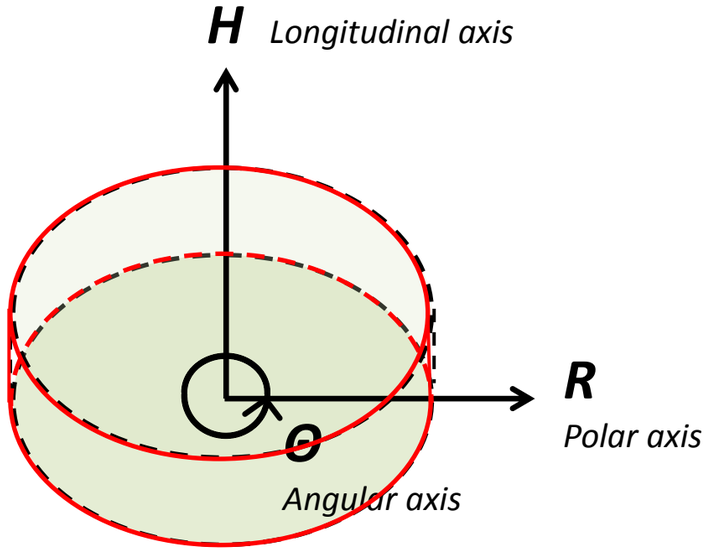




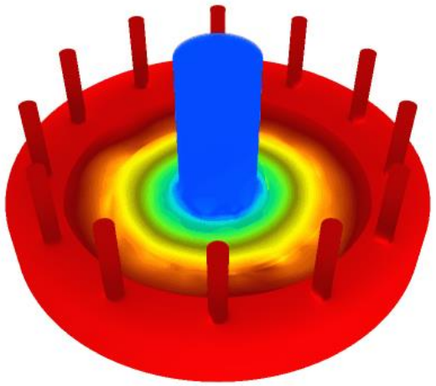
Motivation

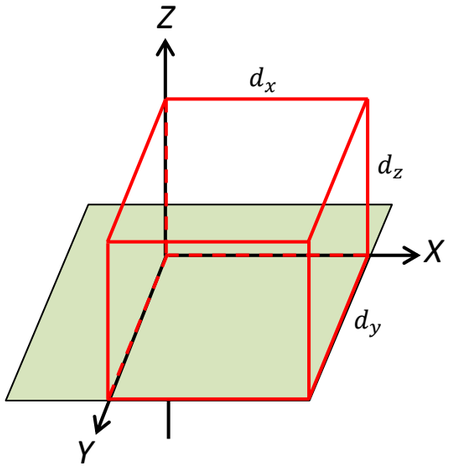


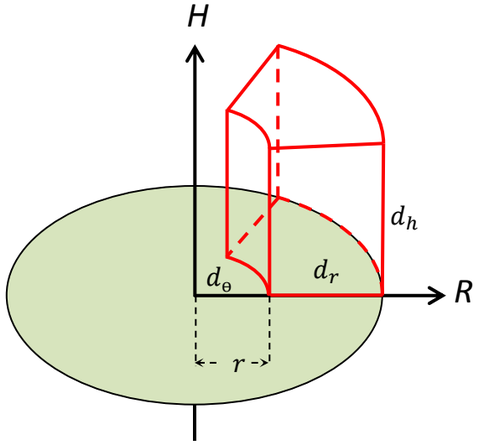




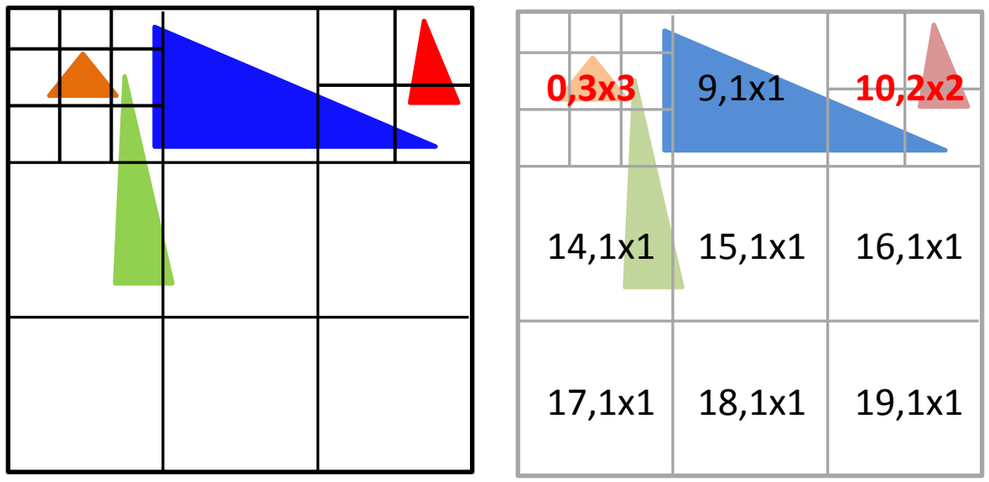
Motivation





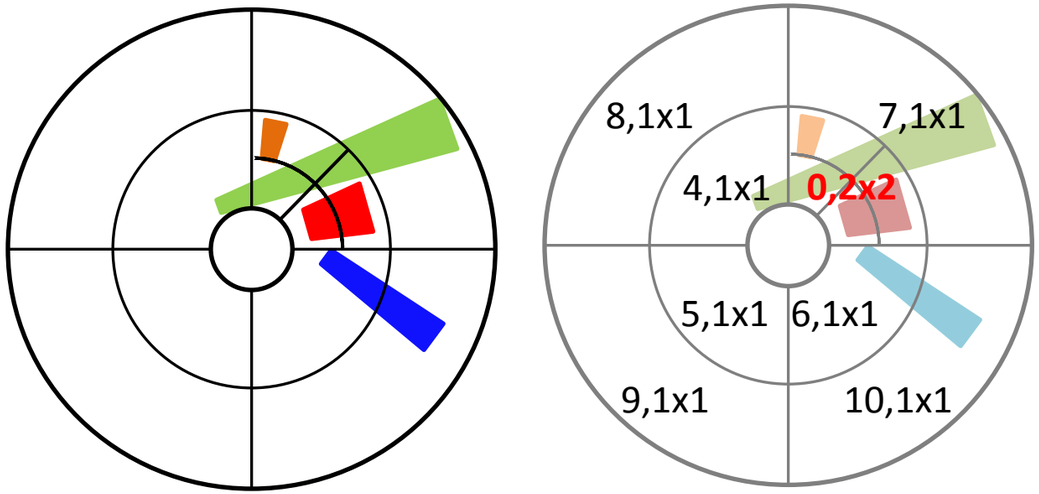


Two-Level Cylindrical Grid



Sort-based Grid Construction
[Kalojanov and Slusallek 2009]

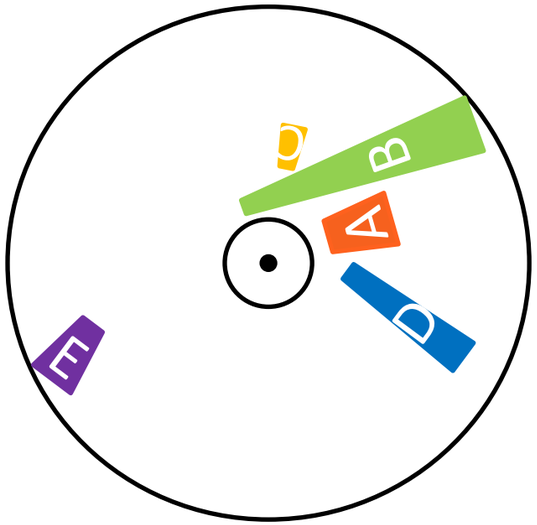
Two-level Cartesian Grid
[Kalojanov et al. 2011]



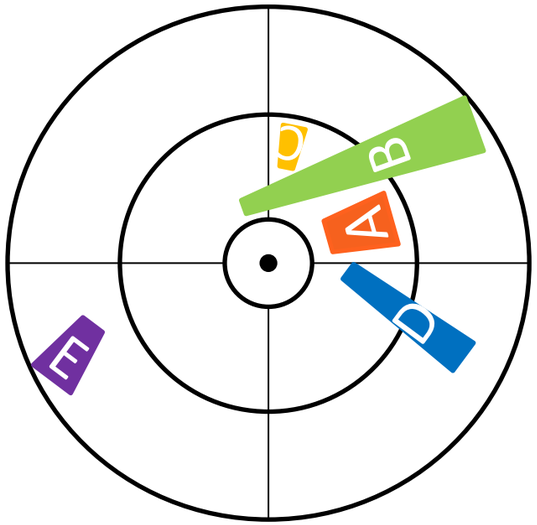
Two-level Cylindrical Grid

Sort-Based Grid Construction

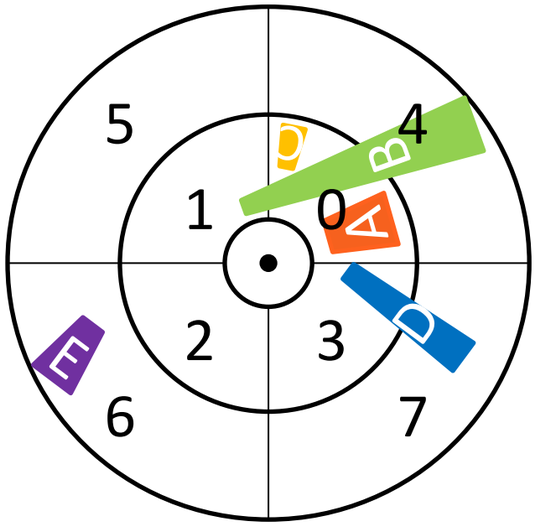
Sort-Based Grid Construction



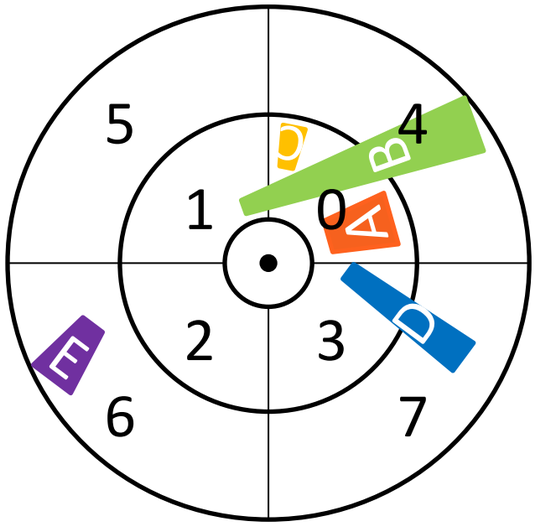
Sort-Based Grid Construction



Sort-Based Grid Construction



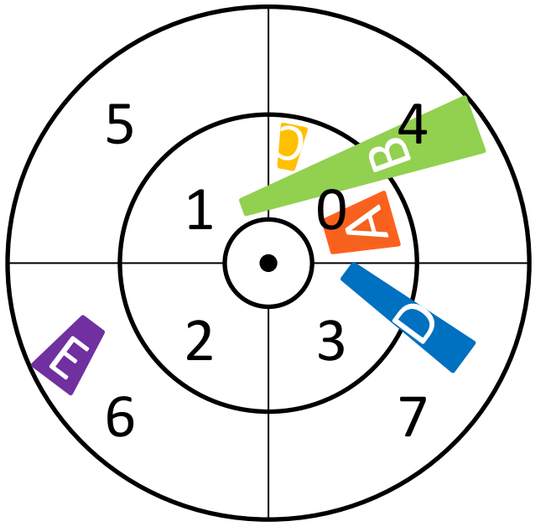
Sort-Based Grid Construction



(cell ID, Prim ID)

0	A

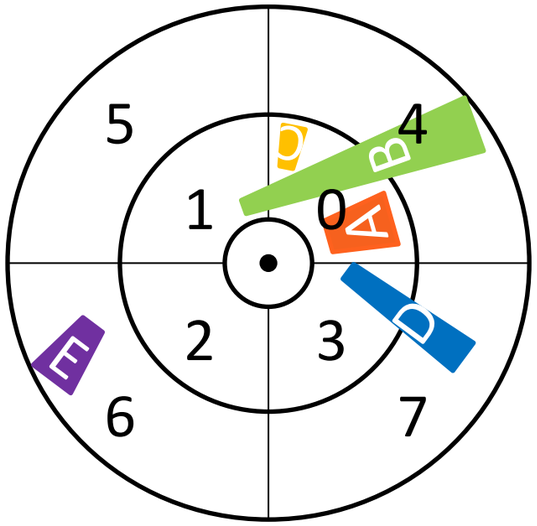
Sort-Based Grid Construction



(cell ID, Prim ID)

0	A	0	B	1	B	4	B

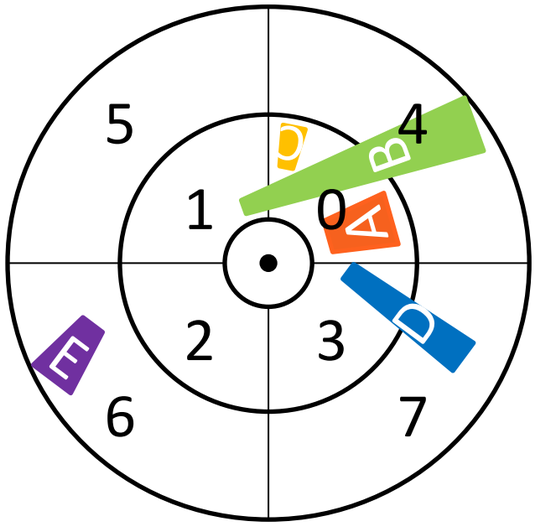
Sort-Based Grid Construction



(cell ID, Prim ID)

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

Sort-Based Grid Construction



(cell ID, Prim ID)

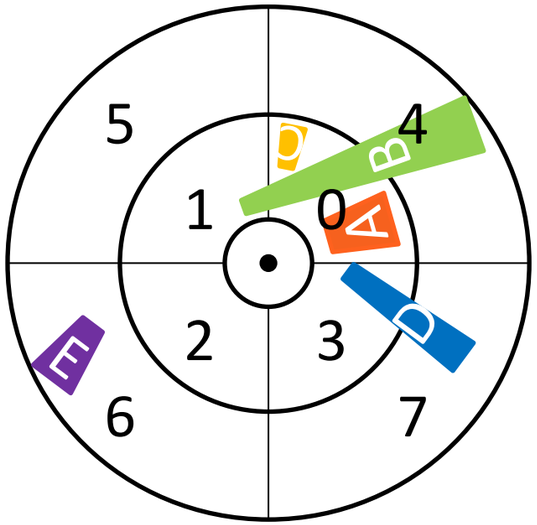
0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

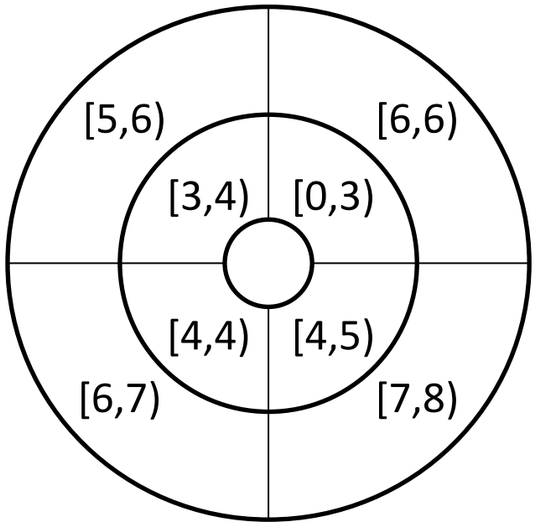
Sort

Reference Array

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Sort-Based Grid Construction





(cell ID, Prim ID)

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

Sort

Reference Array

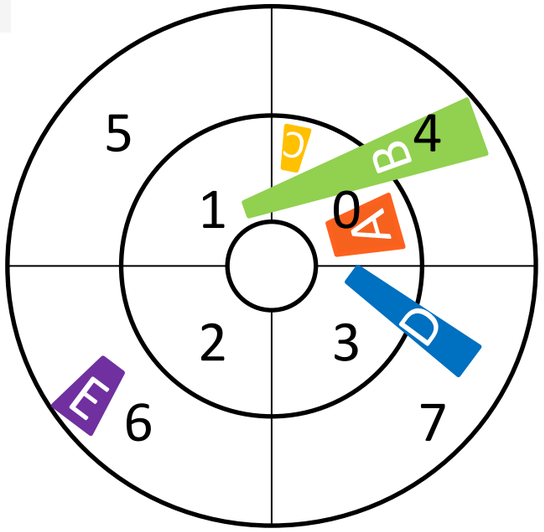
0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Extract Cells

Cell Array

Cell:	0	1	2	3	4	5	6	7
Prim Range:	[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

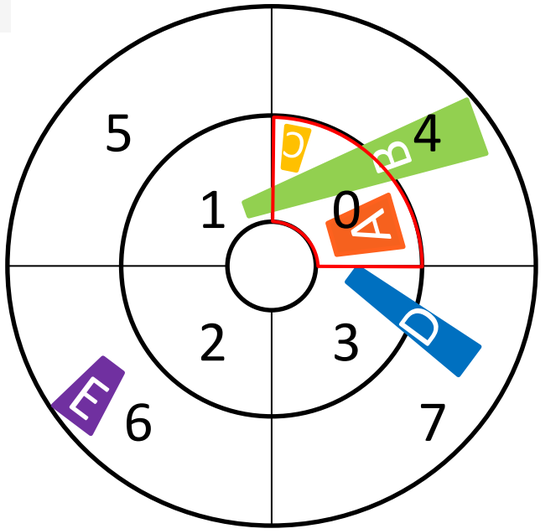
Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

Reference Array — Sort

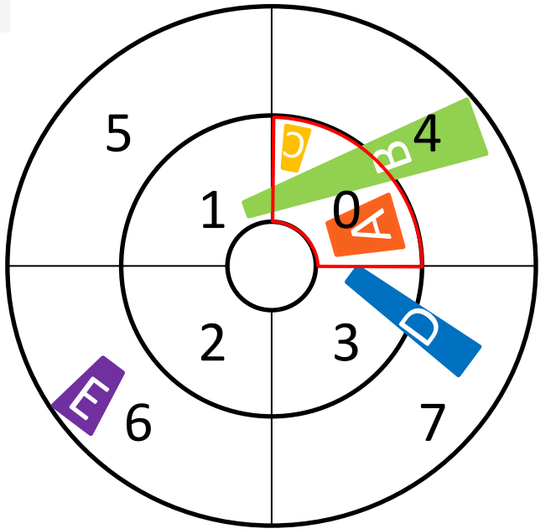
0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

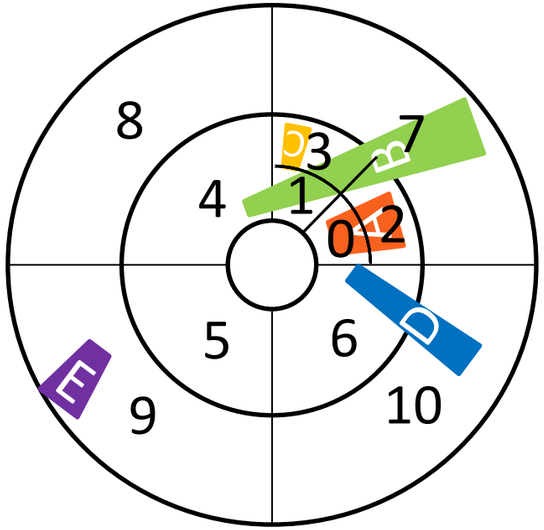
Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid

Sort-Based Two-Level Grid Construction





Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

Reference Array — Sort

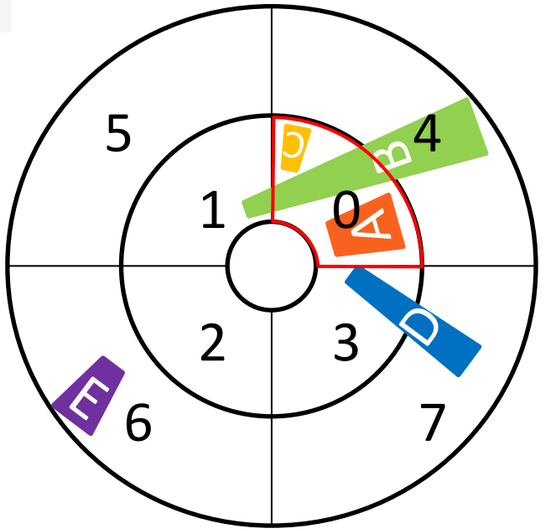
0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

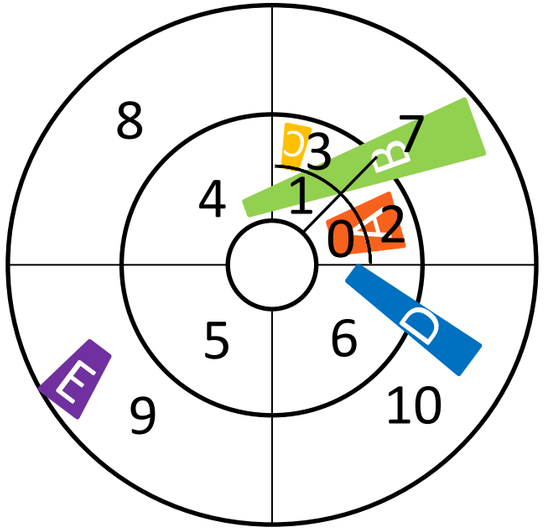
Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

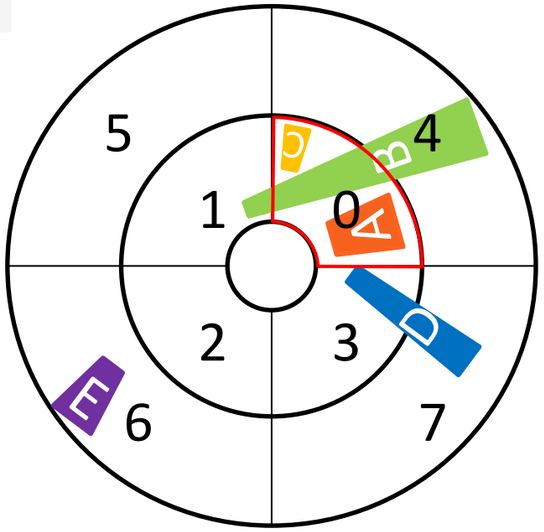
R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid



(cell ID, Prim ID)

0	A	2	A	1	B	2	B	3	B	3	C

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

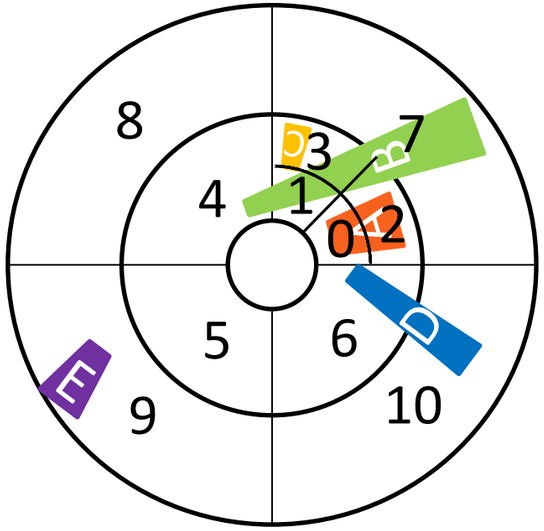
Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid



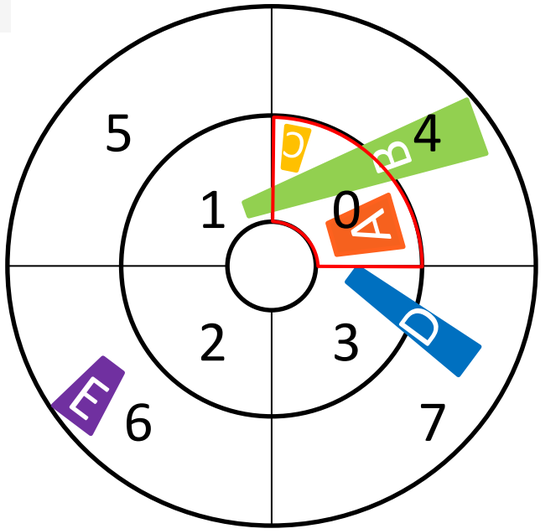
(cell ID, Prim ID)

0	A	2	A	1	B	2	B	3	B	3	C

Reference Array — Sort

0	A	1	B	2	A	2	B	3	B	3	C

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

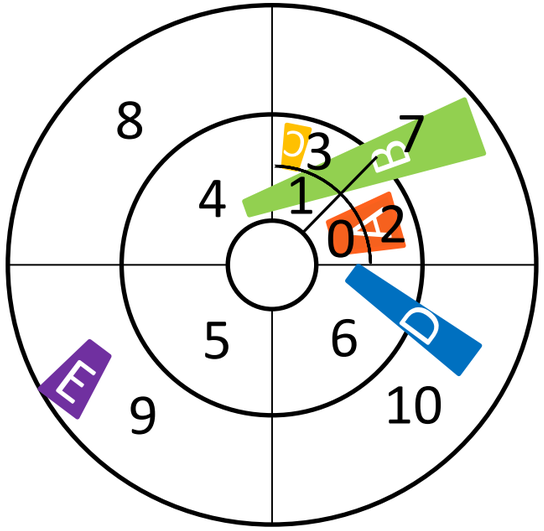
Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid



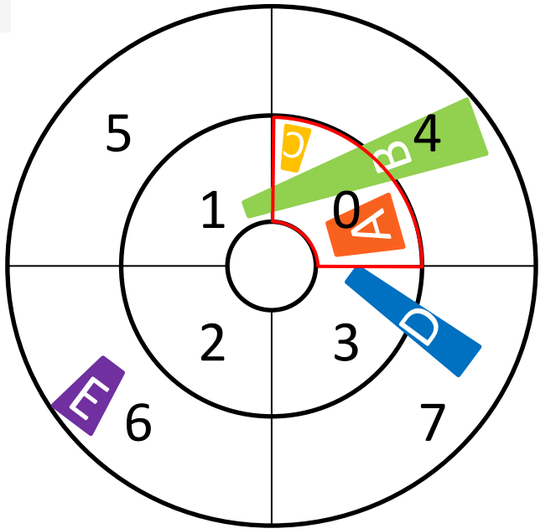
(cell ID, Prim ID)

0	A	2	A	1	B	2	B	3	B	3	C

Reference Array — Sort

0	A	1	B	2	A	2	B	3	B	3	C	4	B	6	D	7	B	9	E	10	D

Sort-Based Two-Level Grid Construction



Top-Level:

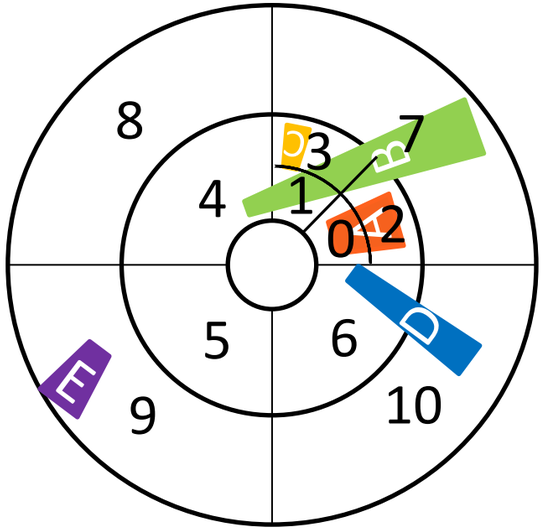
0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)



R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid

(cell ID, Prim ID)

0	A	2	A	1	B	2	B	3	B	3	C

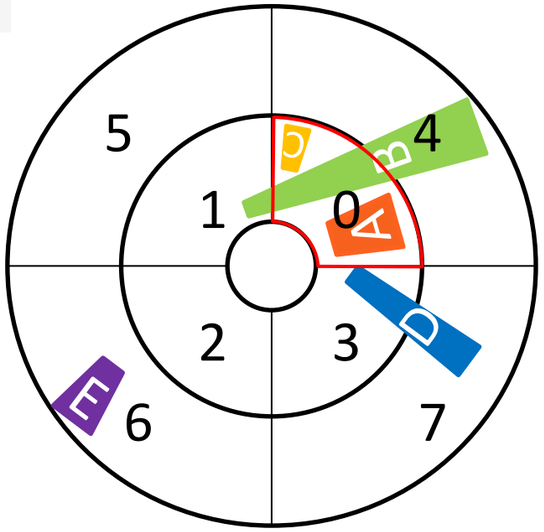
Reference Array — Sort

0	A	1	B	2	A	2	B	3	B	3	C	4	B	6	D	7	B	9	E	10	D

Cell Array (primitive range array) — Extract

[0, 1)	[1, 2)	[2, 4)	[4, 6)	[6, 7)	[7, 7)	[7, 8)	[8, 9)	[9, 9)	[9,10)	[10,11)

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

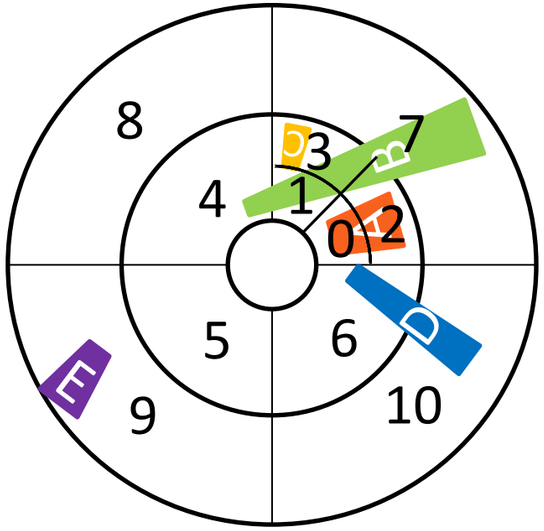
Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid



(cell ID, Prim ID)

0	A	2	A	1	B	2	B	3	B	3	C

Reference Array — Sort

0	A	1	B	2	A	2	B	3	B	3	C	4	B	6	D	7	B	9	E	10	D

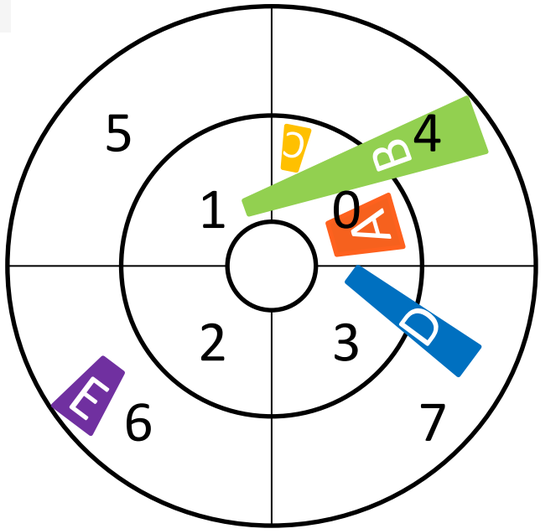
Cell Array (primitive range array) — Extract

[0, 1)	[1, 2)	[2, 4)	[4, 6)	[6, 7)	[7, 7)	[7, 8)	[8, 9)	[9, 9)	[9,10)	[10,11)

Raw Data

A	B	C	D	E

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

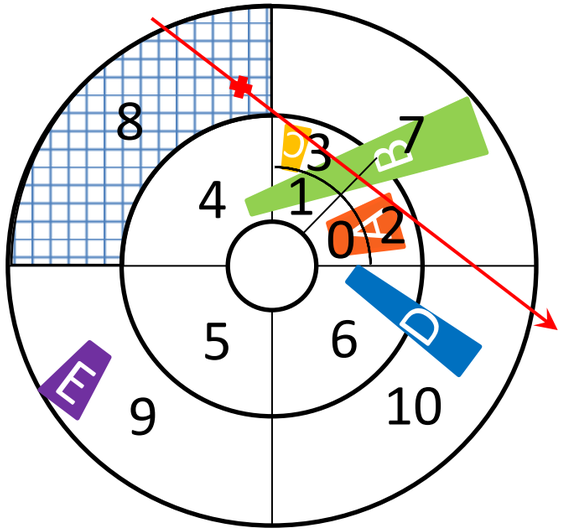
Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid



(cell ID, Prim ID)

0	A	2	A	1	B	2	B	3	B	3	C

Reference Array — Sort

0	A	1	B	2	A	2	B	3	B	3	C	4	B	6	D	7	B	9	E	10	D

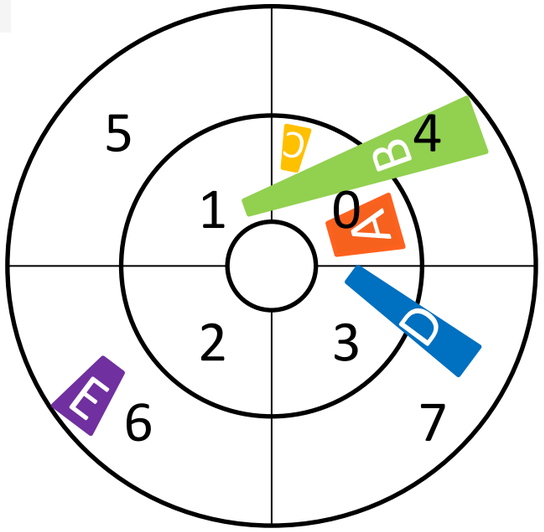
Cell Array (primitive range array) — Extract

[0, 1)	[1, 2)	[2, 4)	[4, 6)	[6, 7)	[7, 7)	[7, 8)	[8, 9)	[9, 9)	[9,10)	[10,11)

Raw Data

A	B	C	D	E

Sort-Based Two-Level Grid Construction



Top-Level:

0	A	0	B	1	B	4	B	0	C	3	D	7	D	6	E

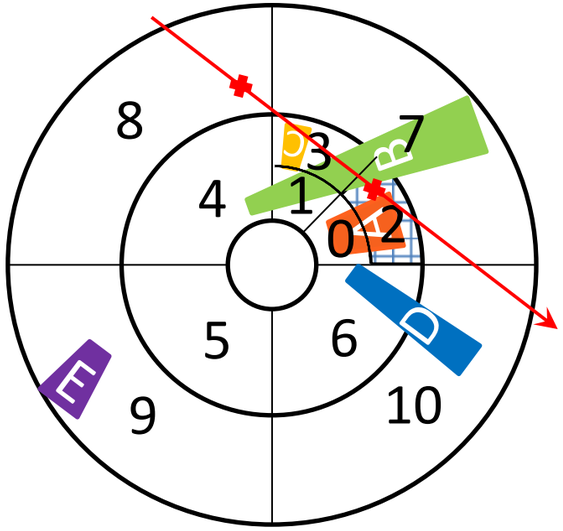
Reference Array — Sort

0	A	0	B	0	C	1	B	3	D	4	B	6	E	7	D

Cell Array — Extract

[0, 3)	[3, 4)	[4, 4)	[4, 5)	[5, 6)	[6, 6)	[6, 7)	[7, 8)

R, Θ, H and N (number of primitives)->Resolution of Second-Level Grid



(cell ID, Prim ID)

0	A	2	A	1	B	2	B	3	B	3	C

Reference Array — Sort

0	A	1	B	2	A	2	B	3	B	3	C	4	B	6	D	7	B	9	E	10	D

Cell Array (primitive range array) — Extract

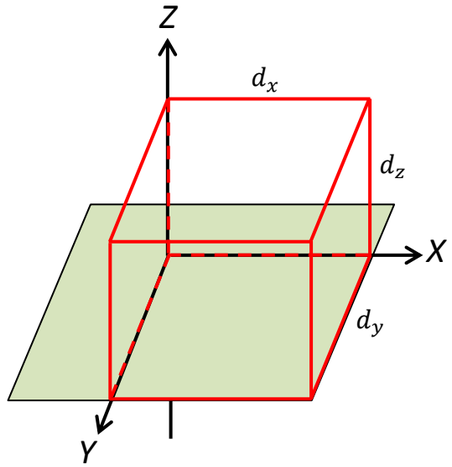
[0, 1)	[1, 2)	[2, 4)	[4, 6)	[6, 7)	[7, 7)	[7, 8)	[8, 9)	[9, 9)	[9,10)	[10,11)

Raw Data

A	B	C	D	E

Sort-Based Two-Level Grid Construction

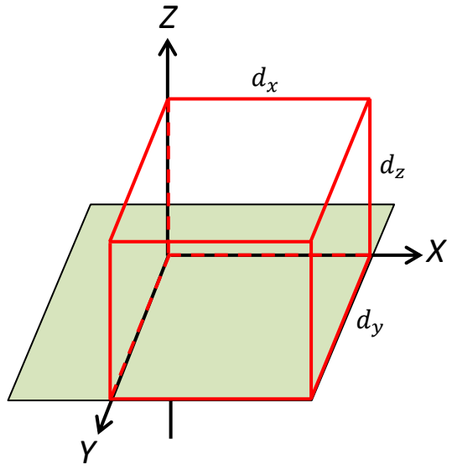
How to derive the grid resolution?



$$R_x = d_x \sqrt[3]{\frac{\lambda N}{V}} \qquad R_y = d_y \sqrt[3]{\frac{\lambda N}{V}} \qquad R_z = d_z \sqrt[3]{\frac{\lambda}{V}}$$

Sort-Based Two-Level Grid Construction

How to derive the grid resolution?

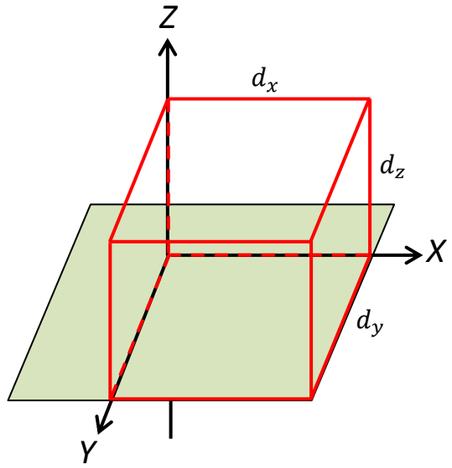


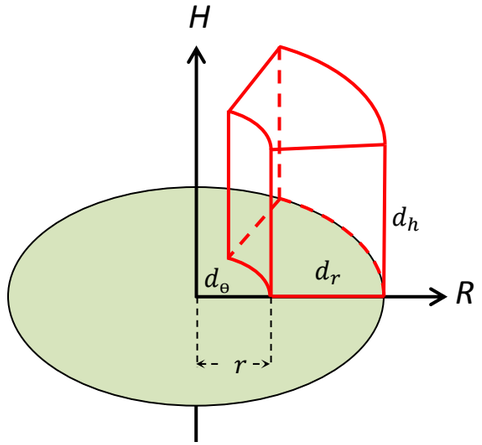
$$R_x = d_x \sqrt[3]{\frac{\lambda N}{V}} \qquad R_y = d_y \sqrt[3]{\frac{\lambda N}{V}} \qquad R_z = d_z \sqrt[3]{\frac{\lambda}{V}}$$

$$\lambda^{-1} = \frac{N}{R_x R_y R_z} \qquad d_x d_y d_z = V$$

Sort-Based Two-Level Grid Construction

How to derive the grid resolution?





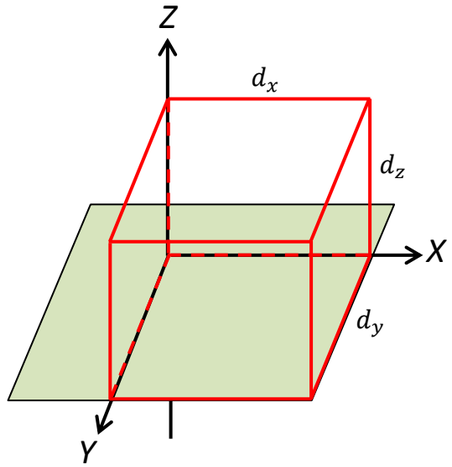
$$R_x = d_x \sqrt[3]{\frac{\lambda N}{V}} \qquad R_y = d_y \sqrt[3]{\frac{\lambda N}{V}} \qquad R_z = d_z \sqrt[3]{\frac{\lambda}{V}}$$

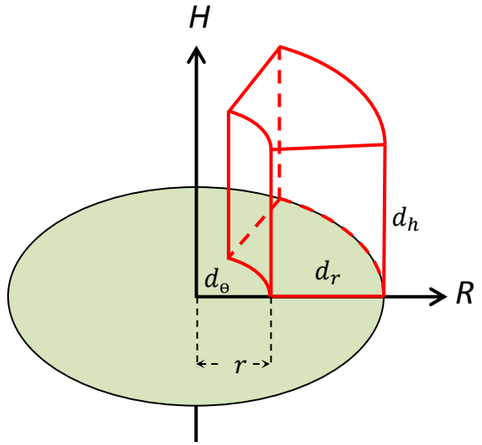
$$\lambda^{-1} = \frac{N}{R_x R_y R_z} \qquad d_x d_y d_z = V$$

$$V = d_r d_\theta \frac{(2r + d_r)}{2} d_h$$

Sort-Based Two-Level Grid Construction

How to derive the grid resolution?





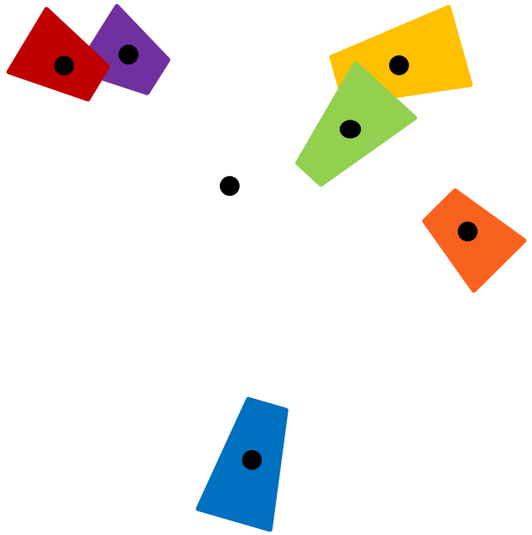
$$R_x = d_x \sqrt[3]{\frac{\lambda N}{V}} \qquad R_y = d_y \sqrt[3]{\frac{\lambda N}{V}} \qquad R_z = d_z \sqrt[3]{\frac{\lambda}{V}}$$

$$\lambda^{-1} = \frac{N}{R_x R_y R_z} \qquad d_x d_y d_z = V$$

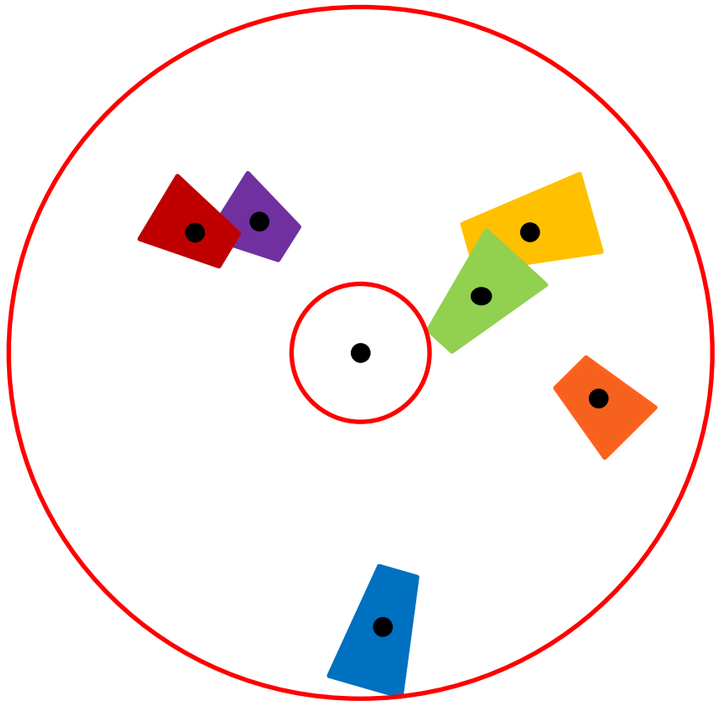
$$V = d_r d_\theta \frac{(2r + d_r)}{2} d_h$$

$$R_r = d_r \sqrt[3]{\frac{\lambda N}{V}} \qquad R_\theta = \frac{d_\theta (2r + d_r)}{2} \sqrt[3]{\frac{\lambda N}{V}} \qquad R_h = d_h \sqrt[3]{\frac{\lambda N}{V}}$$

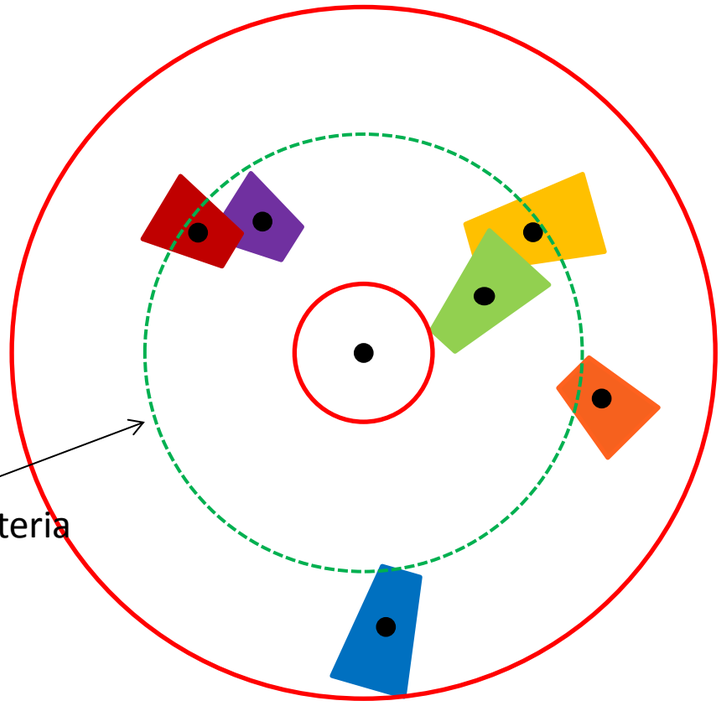
Cylindrical BVH



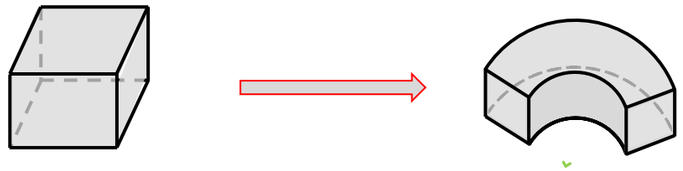
Cylindrical BVH



Cylindrical BVH

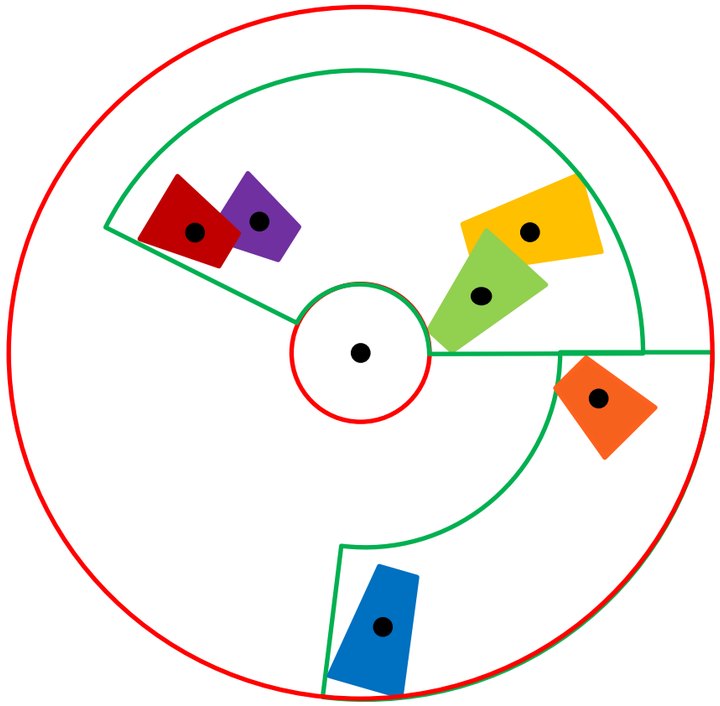


R:

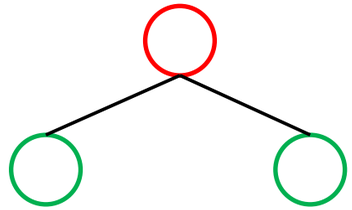


Surface area calculation has changed!

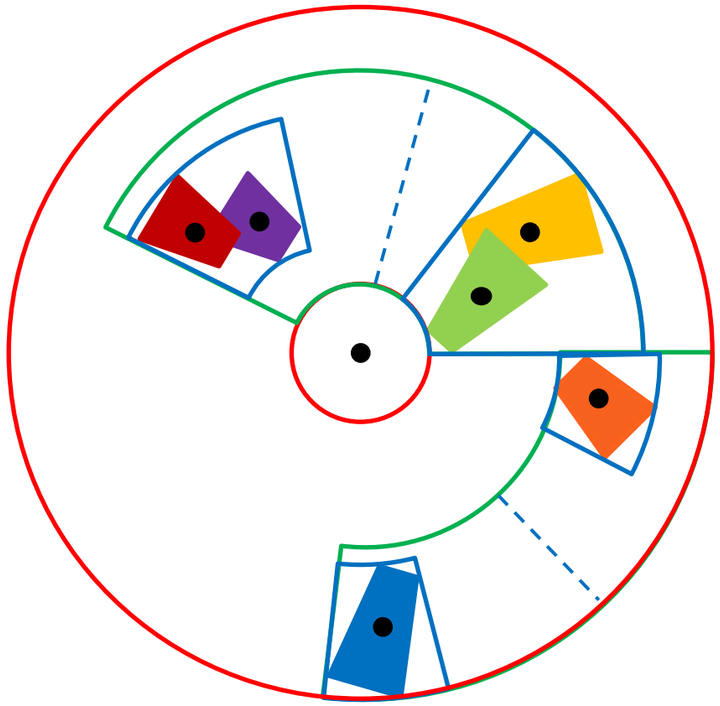
Cylindrical BVH

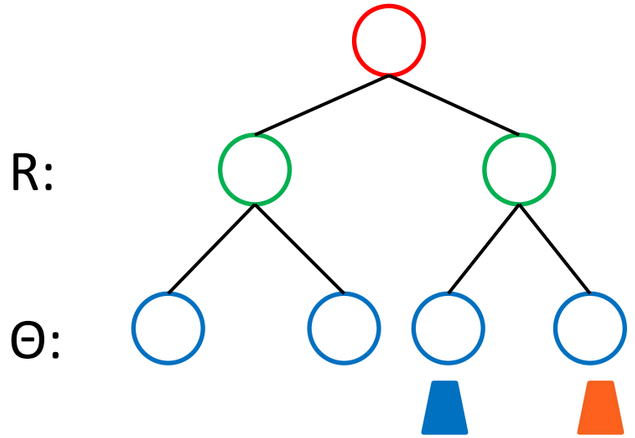


R:

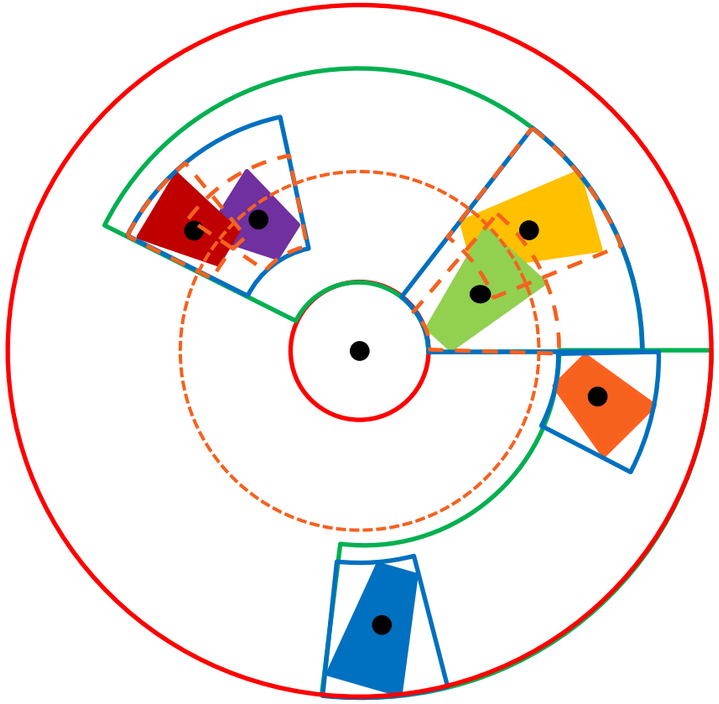


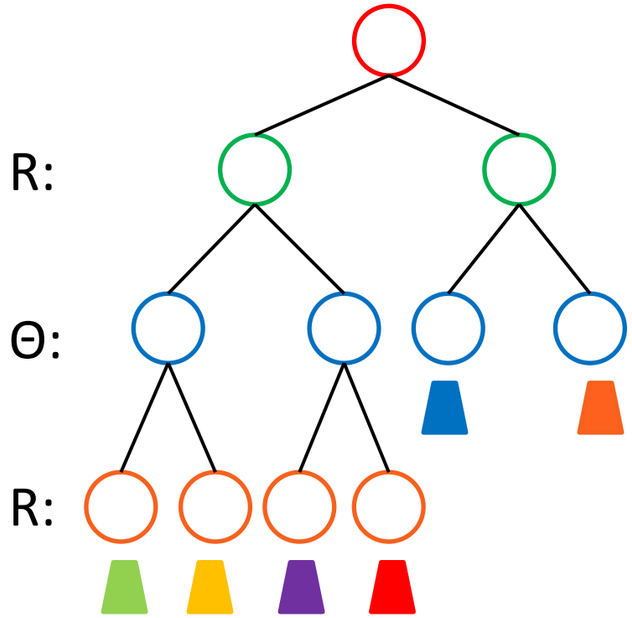
Cylindrical BVH



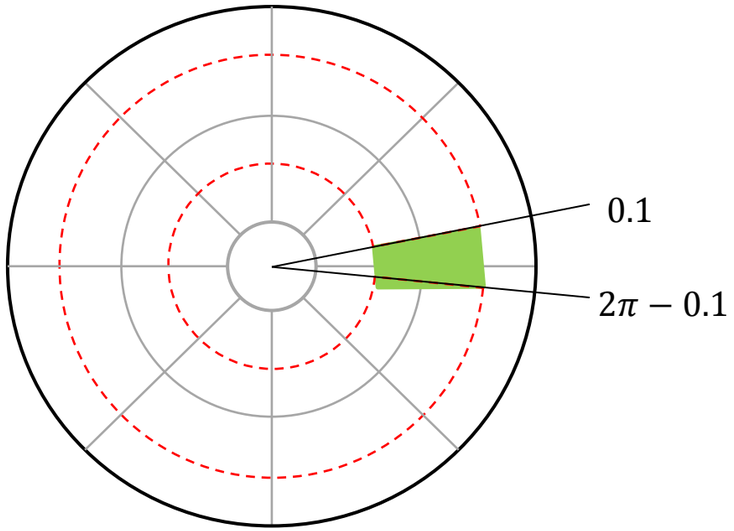


Cylindrical BVH



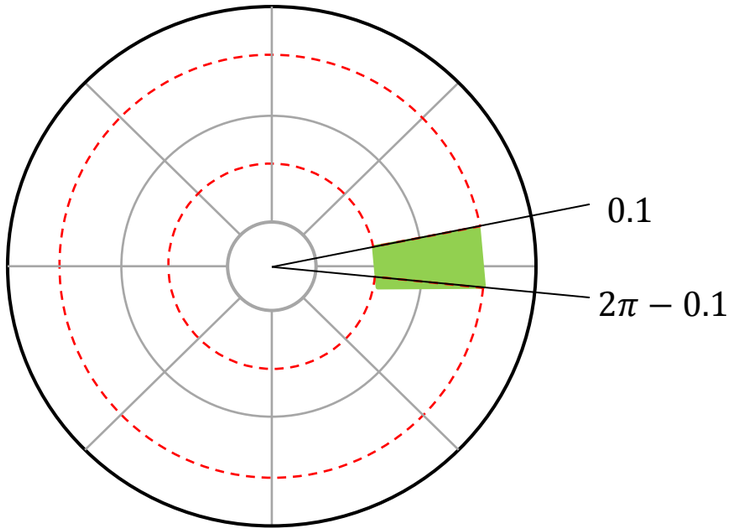


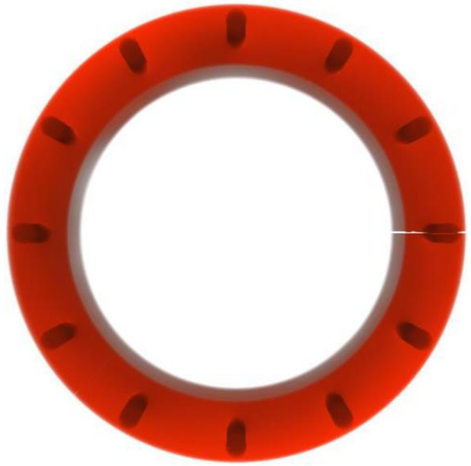
A Special Case

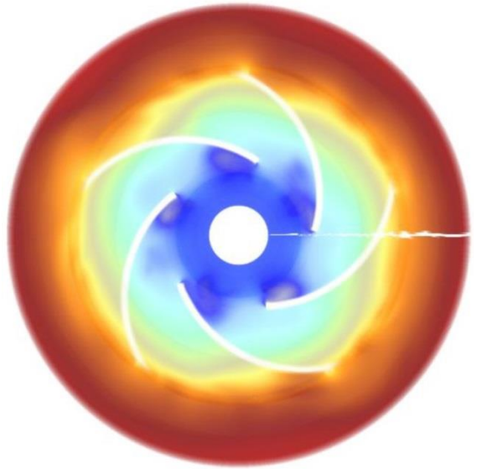


Incorrect bounding volume
Θ dimension is closed and periodical

A Special Case

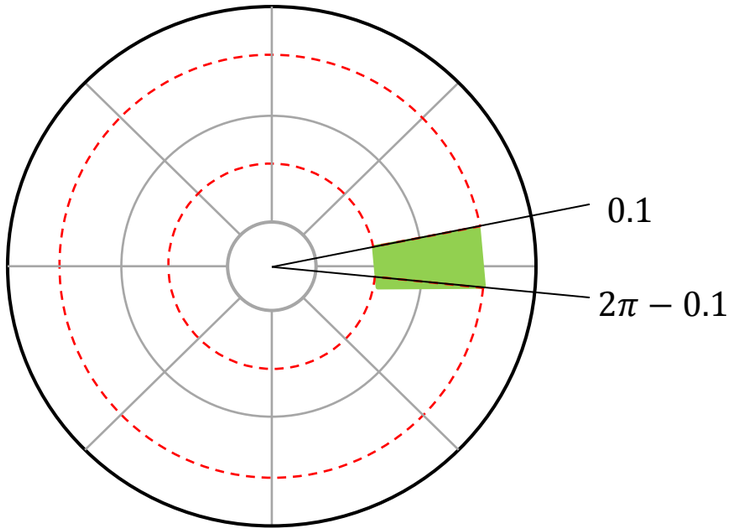




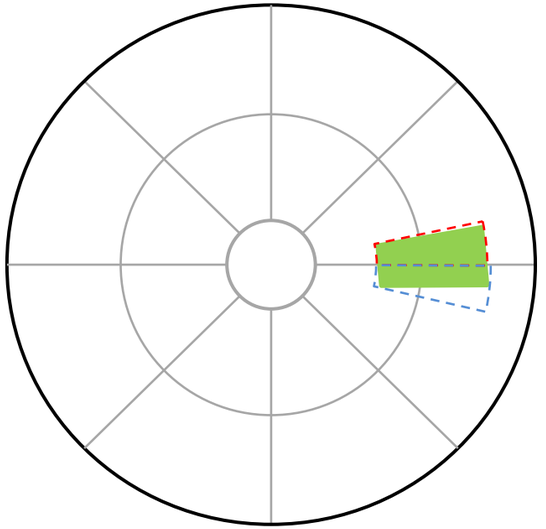


Incorrect bounding volume
Θ dimension is closed and periodical

A Special Case

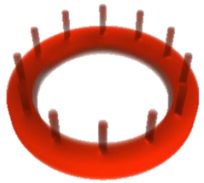
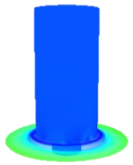
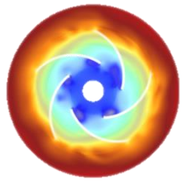
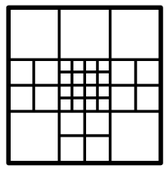
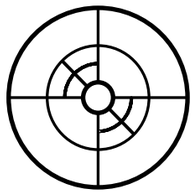


Incorrect bounding volume
Θ dimension is closed and periodical



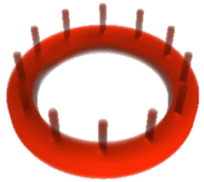
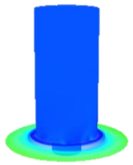
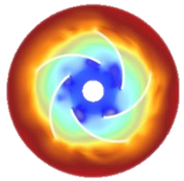
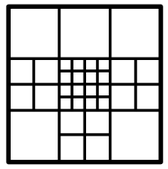
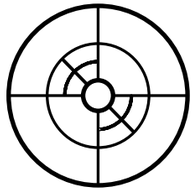
Split hexahedra on the polar axis

Results: Two-Level Grids

Number of Grid Cells	 diffuser	 inlet	 rotor
 Cartesian Grid	2.82×10^7	6.47×10^6	2.39×10^7
 Cylindrical Grid	2.81×10^7	6.94×10^6	2.25×10^7
$Ratio = \frac{Cylindrical}{Cartesian}$	1.00	1.07	0.94

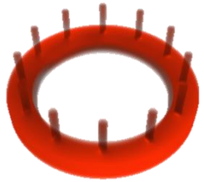
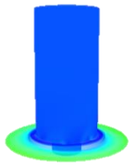
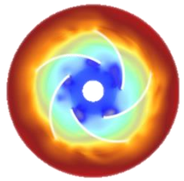
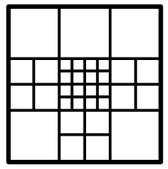
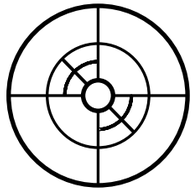
Roughly the same number of grid cells.

Results: Two-Level Grids

Number of Hexahedra	 diffuser	 inlet	 rotor
 Cartesian Grid	2.34×10^8	4.18×10^7	2.29×10^8
 Cylindrical Grid	1.43×10^8	3.09×10^7	1.47×10^8
$Ratio = \frac{Cylindrical}{Cartesian}$	0.61	0.74	0.64

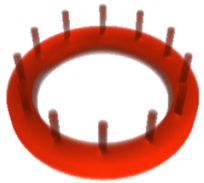
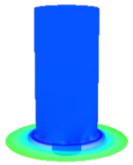
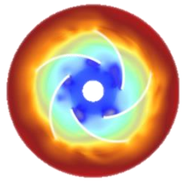
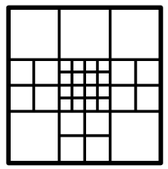
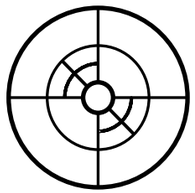
Cylindrical grid generates less duplications.

Results: Two-Level Grids

Memory Usage	 diffuser	 inlet	 rotor
 Cartesian Grid	$1139\ MB$	$214\ MB$	$1084\ MB$
 Cylindrical Grid	$783\ MB$	$175\ MB$	$753\ MB$
$Ratio = \frac{Cylindrical}{Cartesian}$	0.69	0.82	0.69

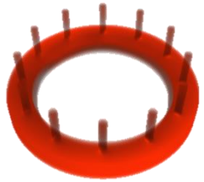
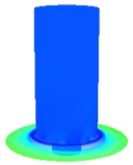
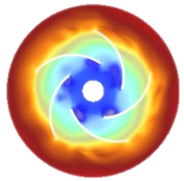
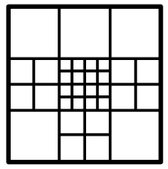
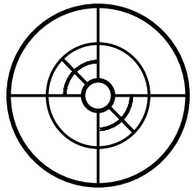
Cylindrical grid uses less memory space.

Results: Two-Level Grids

Hexahedra Visited in Rendering	 diffuser	 inlet	 rotor
 Cartesian Grid	1.66×10^7	2.15×10^7	1.51×10^7
 Cylindrical Grid	9.74×10^6	1.26×10^7	1.35×10^7
$Ratio = \frac{Cylindrical}{Cartesian}$	0.59	0.59	0.89

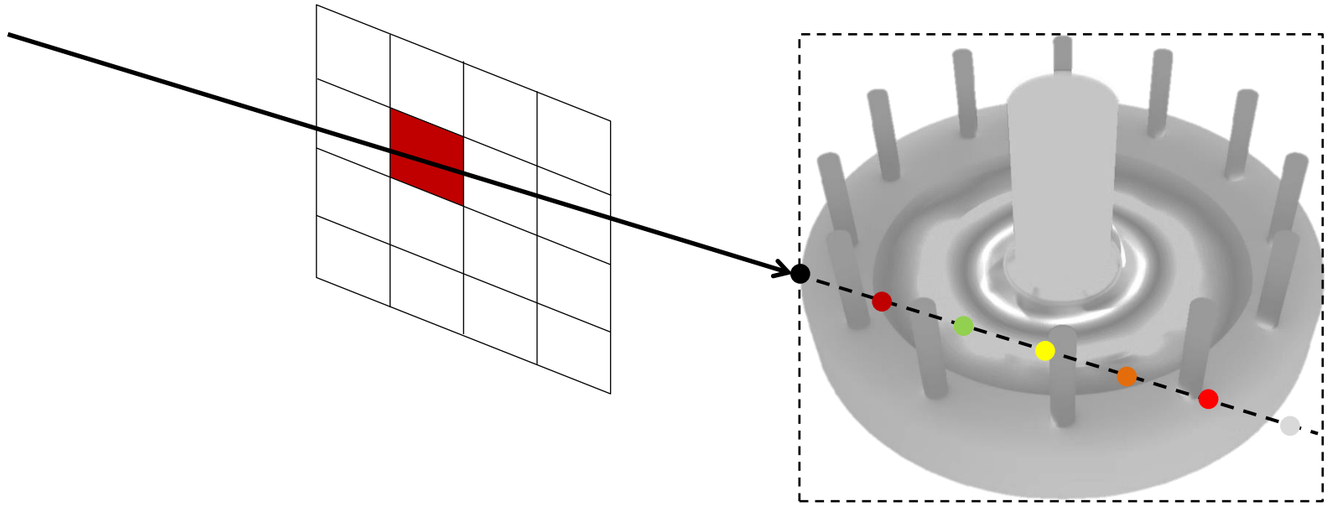
Using cylindrical grid visits less hexahedra.

Results: Two-Level Grids

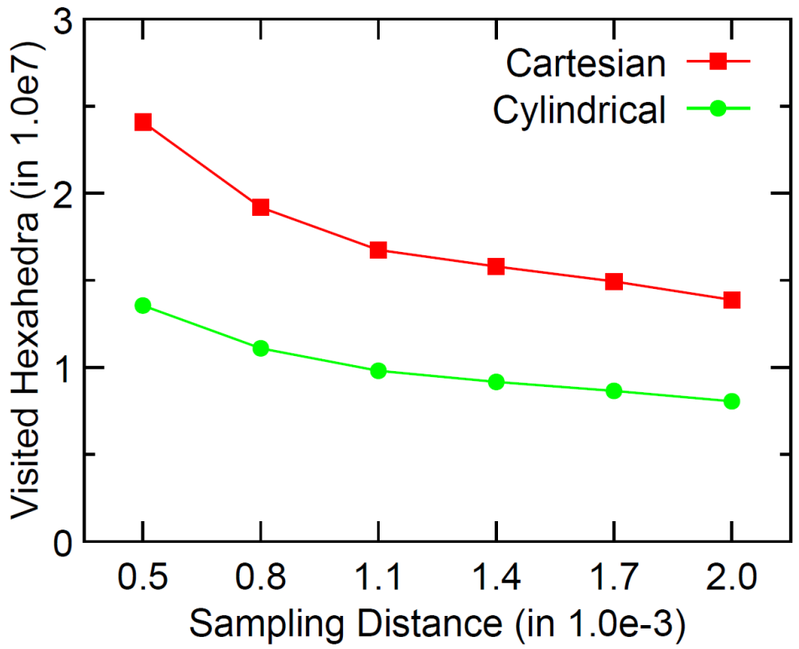
Frame Per Second	 diffuser	 inlet	 rotor
 Cartesian Grid	12.80	14.86	18.61
 Cylindrical Grid	17.39	17.96	20.24
$Ratio = \frac{Cylindrical}{Cartesian}$	1.36	1.21	1.09

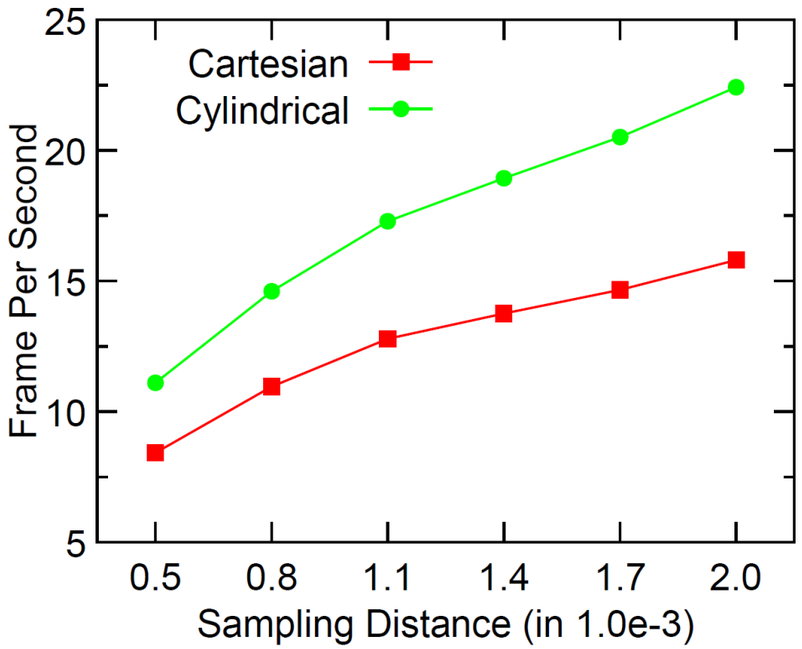
Using cylindrical grid achieves higher frame rates.

Results: Two-Level Grids



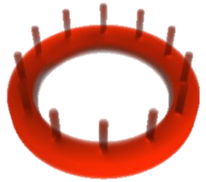
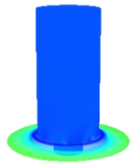
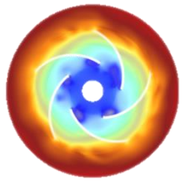
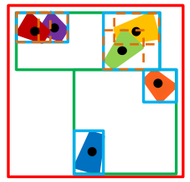
Results: Two-Level Grids

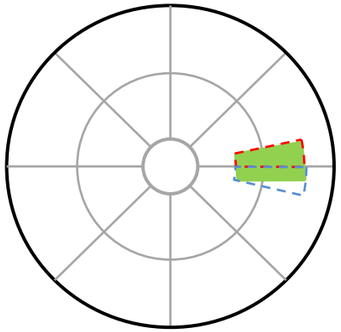




Result of the diffuser domain

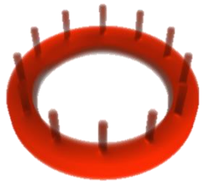
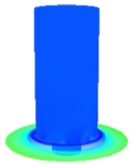
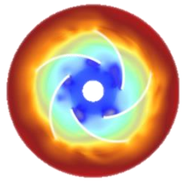
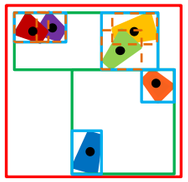
Results: BVH

Number of Hexahedra	 diffuser	 inlet	 rotor
 Cartesian BVH	3.44×10^6	8.56×10^5	2.15×10^6
 Cylindrical BVH	3.45×10^6	8.63×10^5	2.16×10^6
$Ratio = \frac{Cylindrical}{Cartesian}$	1.00	1.01	1.00



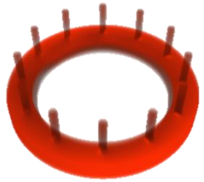
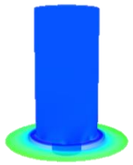
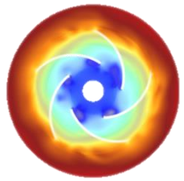
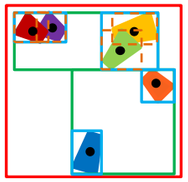
The Cartesian BVH and cylindrical BVH roughly keep the same number of hexahedra.

Results: BVH

Memory Usage	 diffuser	 inlet	 rotor
 Cartesian BVH	$310\ MB$	$77\ MB$	$193\ MB$
 Cylindrical BVH	$311\ MB$	$78\ MB$	$194\ MB$
$Ratio = \frac{Cylindrical}{Cartesian}$	1.00	1.00	1.00

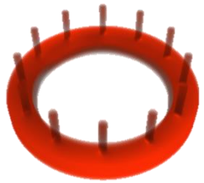
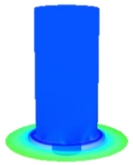
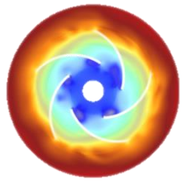
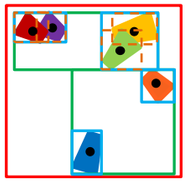
There is no big difference in the memory usage.

Results: BVH

Surface Area	 diffuser	 inlet	 rotor
 Cartesian BVH	87.13	14.21	41.07
 Cylindrical BVH	49.10	10.72	33.01
$Ratio = \frac{Cylindrical}{Cartesian}$	0.56	0.75	0.80

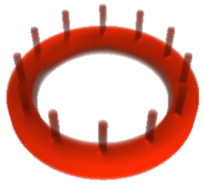
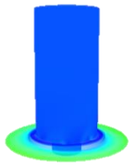
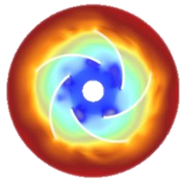
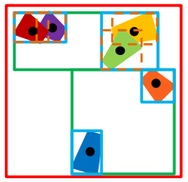
Cylindrical BVH creates tighter bounding volumes for primitives.

Results: BVH

Internal/Leaf Nodes Visited	 diffuser	 inlet	 rotor
 Cartesian BVH	Internal: 2.30×10^8 Leaf: 5.70×10^6	Internal: 2.16×10^8 Leaf: 5.83×10^6	Internal: 9.94×10^7 Leaf: 5.98×10^6
 Cylindrical BVH	Internal: 1.26×10^8 Leaf: 3.64×10^6	Internal: 1.00×10^8 Leaf: 3.97×10^6	Internal: 8.71×10^7 Leaf: 5.15×10^6
$Ratio = \frac{Cylindrical}{Cartesian}$	Internal: 0.55 Leaf: 0.64	Internal: 0.46 Leaf: 0.68	Internal: 0.88 Leaf: 0.86

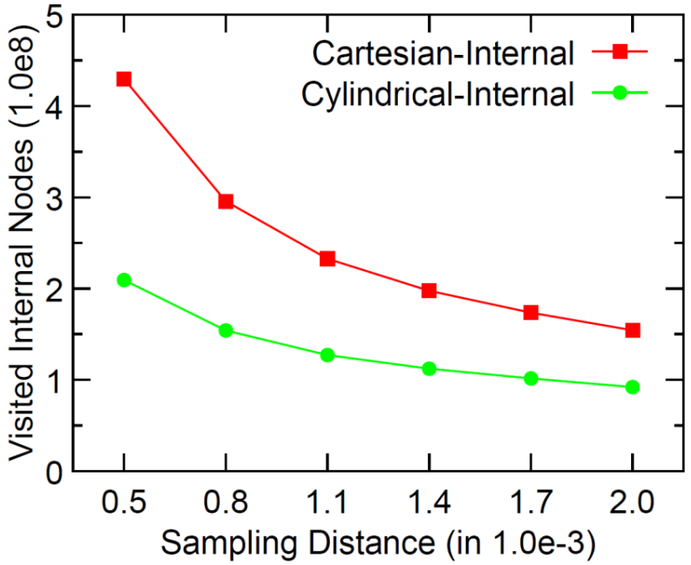
Using cylindrical BVH will visit less Internal/leaf nodes during rendering.

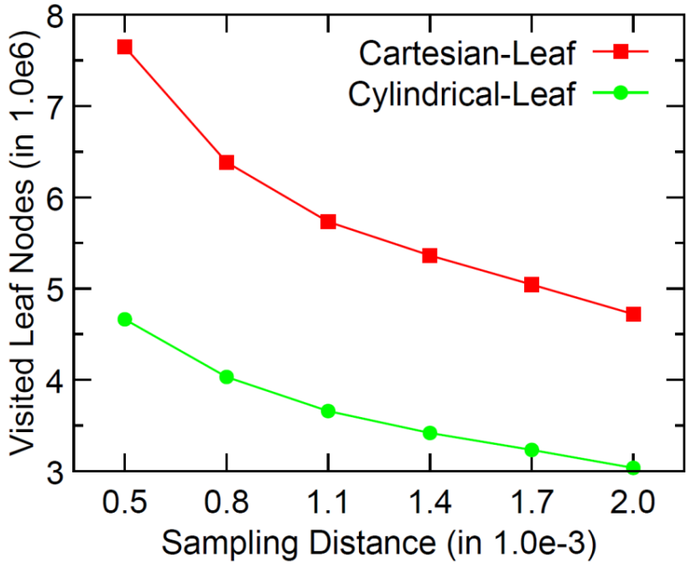
Results: BVH

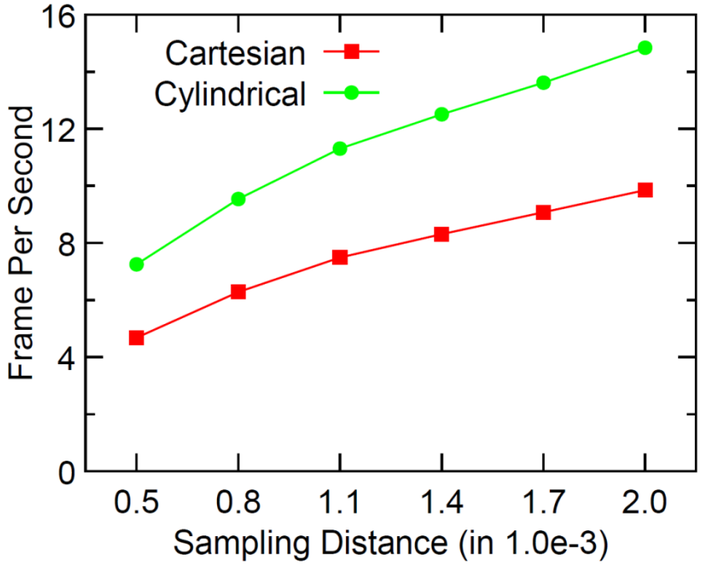
Frame Per Second	 diffuser	 inlet	 rotor
 Cartesian BVH	7.55	9.50	12.16
 Cylindrical BVH	11.37	16.53	13.92
$Ratio = \frac{Cylindrical}{Cartesian}$	1.51	1.74	1.14

Using cylindrical BVH achieves higher frame rates.

Results: BVH







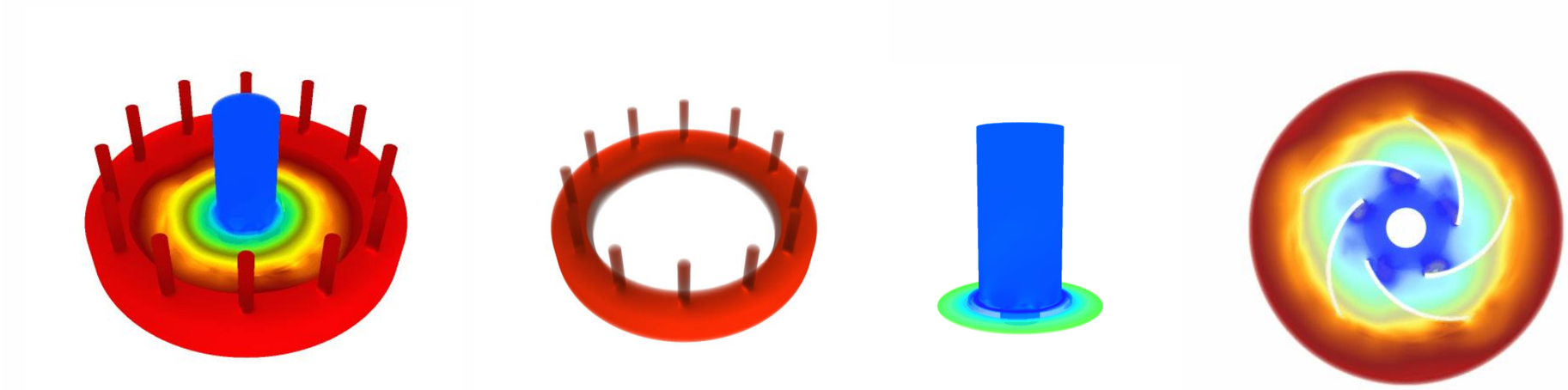
Result of the diffuser domain

Conclusion

- We present two cylindrical acceleration data structures for irregular hexahedral volume visualization. The cylindrical structures outperform their Cartesian counterparts for cylindrical volume data:
  - Two-level Grid: Use less memory (less duplication), visit less hexahedra during traversal
  - BVH: Tighter bounding box (less surface area), visit less BVH internal/leaf nodes

- We hope that the results we were able to achieve would inspire more research and better customizations of acceleration structures to meet the increasingly complex requirements of the data.

Thank you



Questions?