Geometric Partitioning for Tomography

Jan-Willem Buurlage, CWI Amsterdam
Rob Bisseling, Utrecht University
Joost Batenburg, CWI Amsterdam

2018-09-03, Tokyo, Japan
SIAM Conference on Parallel Processing for Scientific Computing

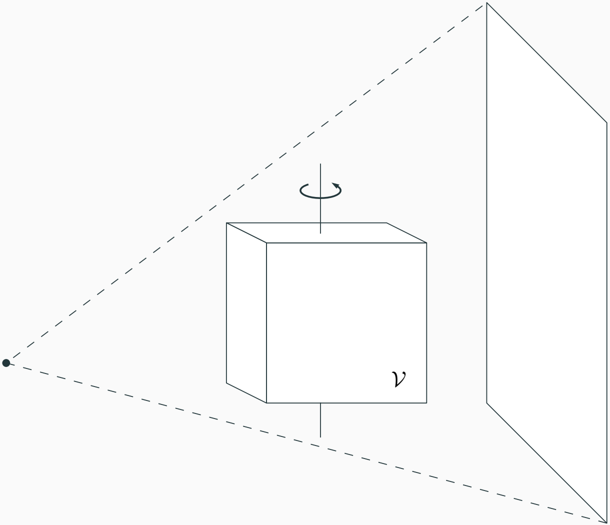
Outline

1. Tomography and tomographic reconstruction
2. Partitioning for distributed tomography
3. Geometric recursive coordinate bisectioning (GRCB)
4. Results and conclusion

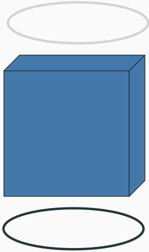
Tomography

- **Tomography** is a non-destructive imaging technique
- Penetrating rays (e.g. X-rays) are sent through an object from various angles, and their intensity is measured
- Leads to 2D projection images, from which a 3D volume is reconstructed

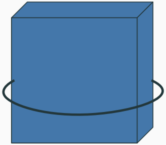
Example of tomographic measurement



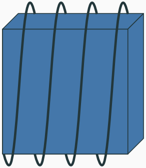
Acquisition geometries



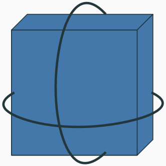
Laminography



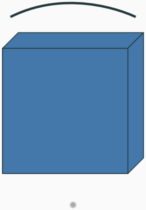
Single axis



Helical cone beam



Dual axis



Tomosynthesis

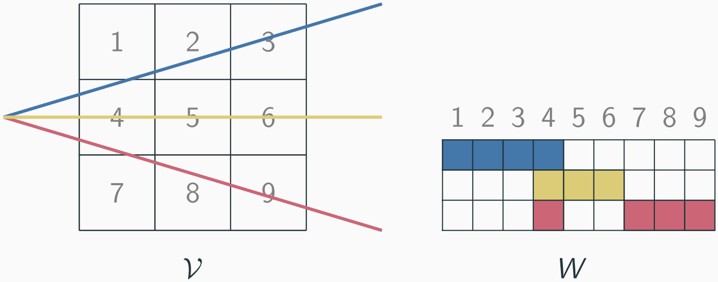
Tomographic reconstruction

- *Projection matrix* W, solve:

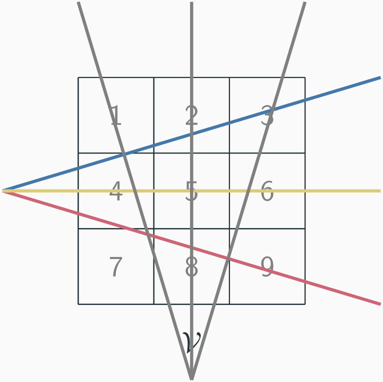
$$W\mathbf{x} = \mathbf{b},$$

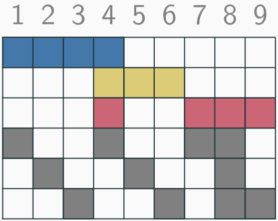
  with $\mathbf{x}$ the *image*, and $\mathbf{b}$ the *projection data*.
- Rows correspond to rays, from a source to a detector pixel. Columns correspond to volume elements, or voxels.
- Intersections of rays with voxels, give rise to nonzeros in W.
- *Note*: W is *sparse*, for n voxels we have $\mathcal{O}(n^{1/3})$ nonzeros in each row.

Example of projection matrix (2D) (I)



Example of Projection Matrix (2D) (II)





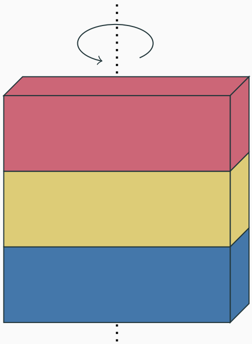
W

Large-scale tomography

- For *simultaneous iterative* reconstruction, the SpMVs $W\mathbf{x}$ and $W^T\mathbf{y}$ are the most expensive operations.
- 3D volumes with at least 1000^3 voxels. W then has $\geq \mathcal{O}(10^{12})$ entries $\Rightarrow$ TBs of data!
- Not stored explicitly, generated from the acquisition geometry.

Large-scale tomography (cont.)

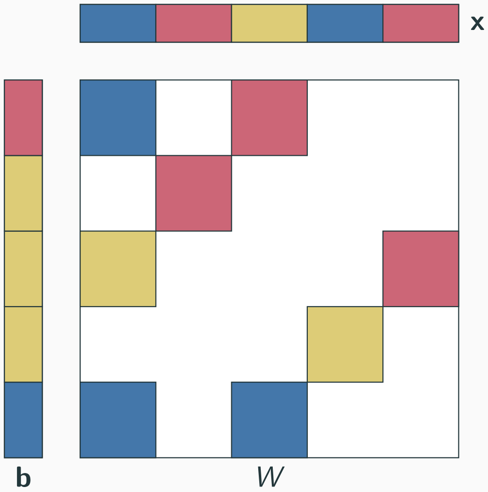
- We parallelize the forward projection and backward projection.
- How to distribute W? Current practice (slabs) leads to prohibitively large communication volumes.
- Available sparse matrix partitioning methods do not scale, since the matrix cannot be stored explicitly.



Parallel SpMV

- When performing an SpMV in parallel, we distribute the data $(W, \mathbf{x}, \mathbf{b})$ over processing elements.
- The distribution of the nonzeros of W are leading; the distribution of $\mathbf{x}$ and $\mathbf{b}$ follow.
- Two types of partitionings:
  - assign rows, or columns, to a single processor (1D partitioning).
  - treat all nonzeros independently (2D partitioning).
- **Warning:** 1D partitioning $\rightarrow$ 3D partitioning in space

Distribution example



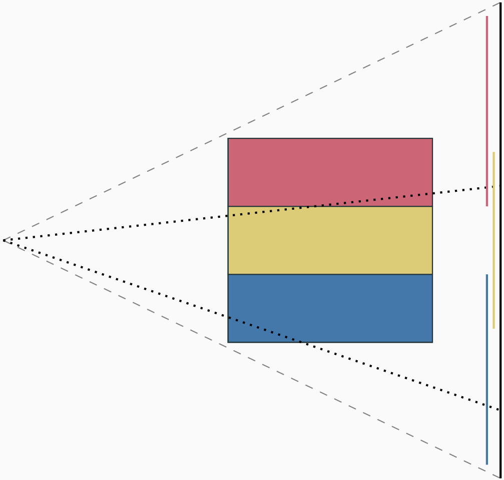
Geometric partitioning

- We exploit the geometric structure of the problem to find a partitioning[1].
- Generate a 3D cuboid partitioning of the object volume, corresponding to a 1D column partitioning of the matrix.
- The communication volume is equal to the total line cut, the number of parts crossed by a ray.

[1] *A geometric partitioning method for distributed tomographic reconstruction*, JB, Rob H. Bisseling, K. Joost Batenburg *(under revision)*

Example of line cut (2D)

- Overlapping *shadows* of subvolumes on the detector show which pixels define cut lines

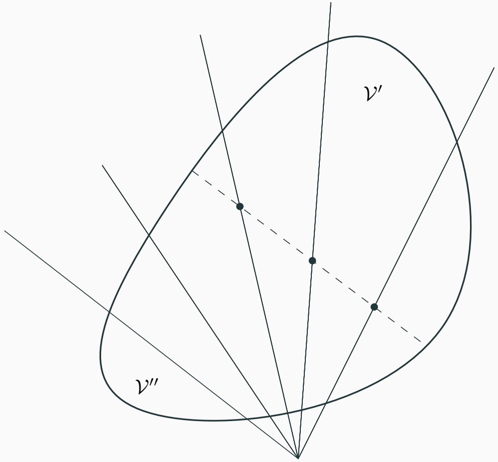


Recursive bisectioning

- **Idea:** Split the volume into two subvolumes recursively.
- Straightforward to show that this can be done independently from previous splits.
- When splitting a subvolume, the effect on the overall communication volume is the same as that of the subproblem.

Interface intersection (2D)

- Communication volume equals number of lines through interface



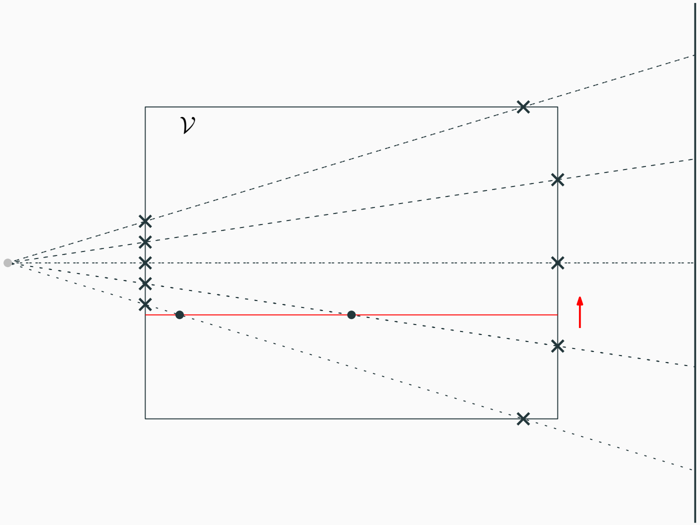
Bisectioning algorithm

- Choose the splitting interface with the minimum number of rays passing through it.
- Evenly distribute the workload
- Computational weight of a voxel is the number of lines crossing the voxel, i.e. number of nonzeros in its column
- Total computational weight of a subvolume can be computed using 3D prefix sums and application of inclusion-exclusion principle.

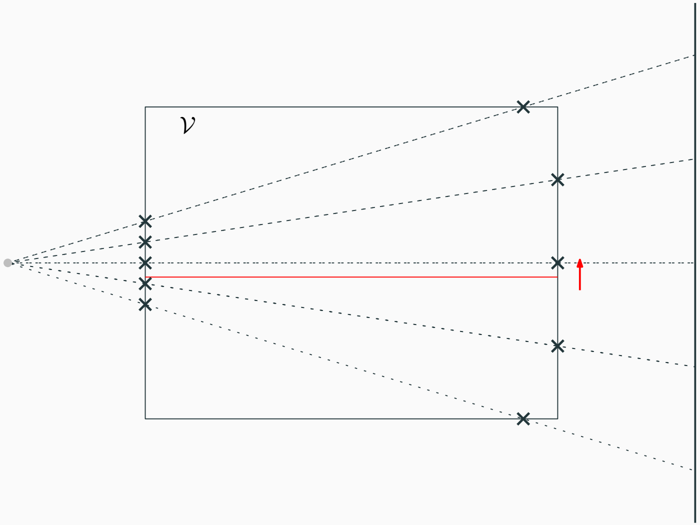
Plane sweep

- We sweep a candidate interface along the volume, and keep track of the current number of rays passing through it.
- *Communication volume only changes at coordinates where a ray intersects the boundary!*
- Compute intersections once, sweep for all three axes sorting the coordinates each time.

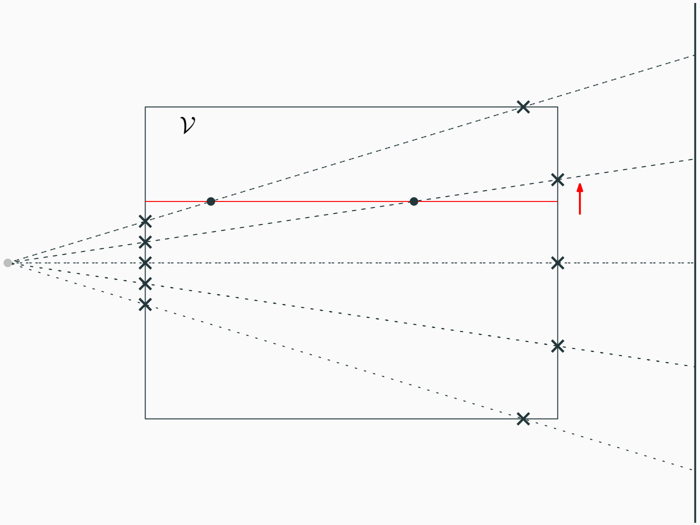
Example of plane sweep (2D) (I)



Example of plane sweep (2D) (II)



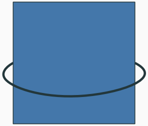
Example of plane sweep (2D) (III)

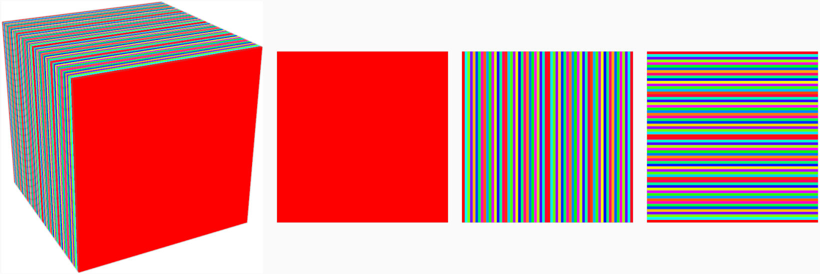


Results

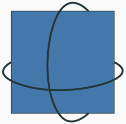
- This gives us an efficient partitioning algorithm, runtime dominated by the sorting of coordinates: $\mathcal{O}(m \log(m))$.
- Geometric recursive coordinate partitioning (GRCB).
- Currently, slab partitionings of the volume along the rotation axis are used.

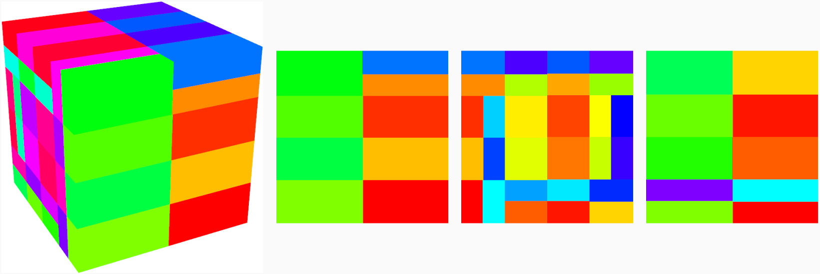
Results (Single-axis parallel beam)



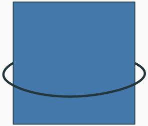


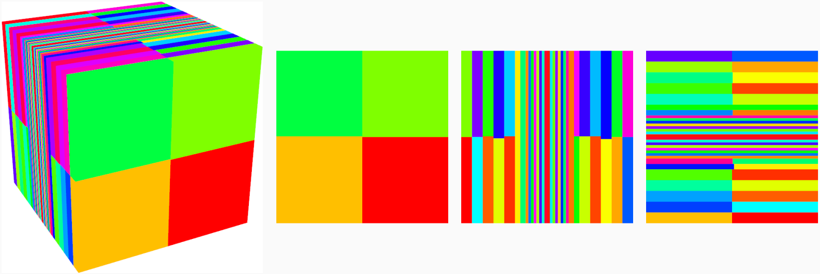
Results (Dual-axis parallel beam)



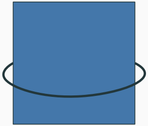


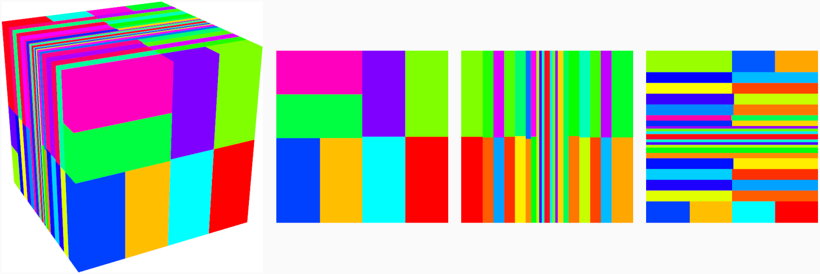
Results (Cone beam with narrow angle)



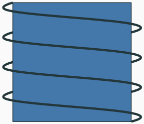


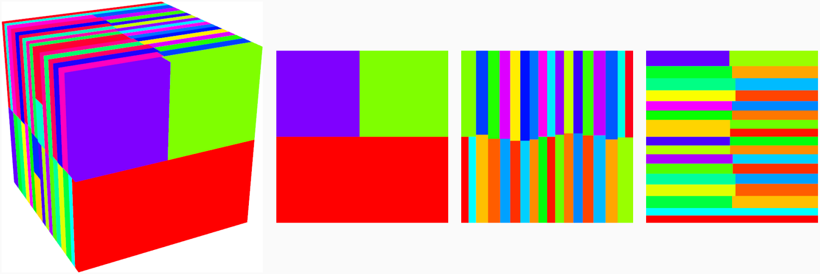
Results (Cone beam with wide angle)



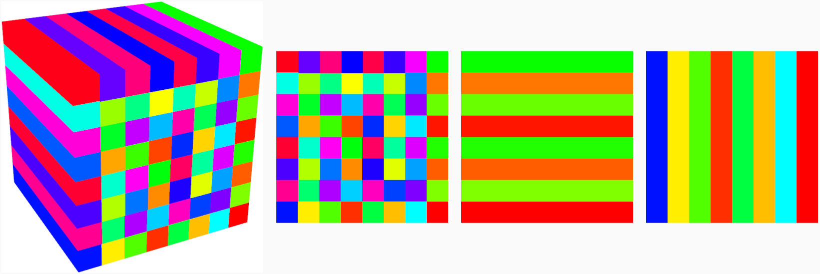


Results (Helical cone beam)

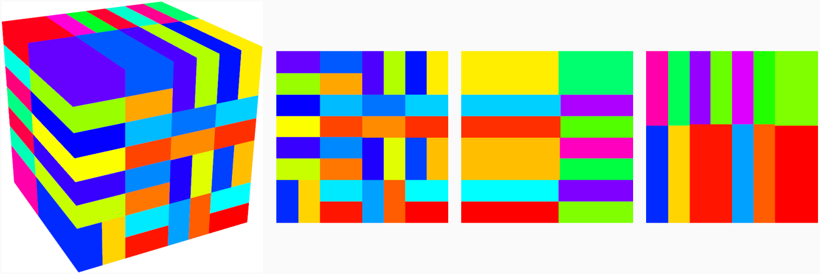




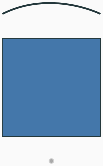
Results (Laminography with narrow angle)

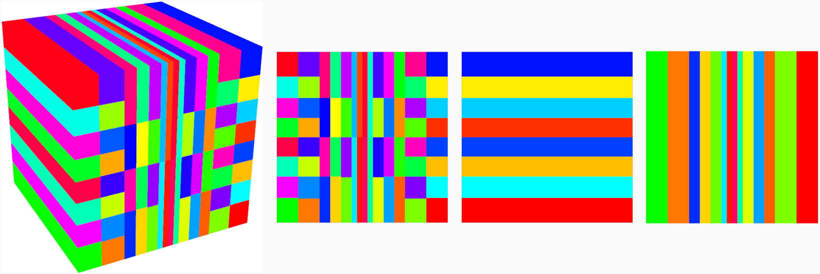


Results (Laminography with wide angle)



Results (Tomosynthesis)





Bulk

- **Bulk**[2] is a BSP library for modern C++
- Provides a safe and simple layer on top of low-level technologies, such as C++ threads or MPI
- Unified and *modern* interface for distributed and parallel computing.

```
auto q = bulk::queue<int, T>(world);
for (auto [target, local, remote] : shared_pixels) {
    q(target).send(remote, projs[local]);
}
```

[2] https://jwbuurlage.github.io/Bulk

Results (Communication volume)

- Results for $p = 256$

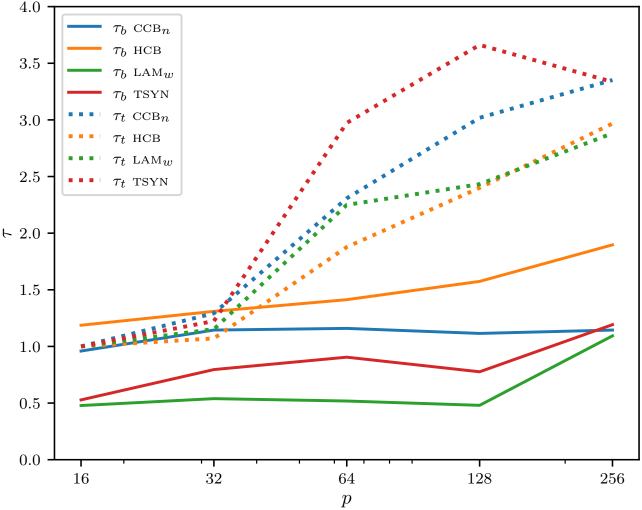
Geometry	V (slab)	V (GRCB)	Improvement
SAPB	0	0	0%
DAPB	1×10^{10}	8×10^{8}	92%
CCBn	1×10^{9}	3×10^{8}	69%
CCBw	2×10^{9}	4×10^{8}	82%
HCB	2×10^{9}	4×10^{8}	71%
LAMn	3×10^{9}	4×10^{8}	89%
LAMw	5×10^{9}	6×10^{8}	90%
TSYN	2×10^{9}	3×10^{8}	87%

GRCB vs Mondriaan

p	1D block	GRCB	Mondriaan
16	111248	111207	108741
32	233095	216620	210330
64	3928222	2505646	2604930

- GRCB versus Mondriaan (1D column) with medium-grain splitting strategy
- Cone beam narrow for 128^3 voxels with 128 projections of size 128×128

Results (Communication time)



Conclusion

- Distributed-memory methods for tomographic reconstruction come with a challenging partitioning problem.
- We present **GRCB**: an efficient and effective partitioning method that leads to low communication volumes and good load balance.