Centrum Wiskunde & Informatica

Sequential sparse matrix–vector multiplication and tomography

Jan-Willem Buurlage (CWI)

MasterMath: Parallel Computing (2018)

CWI

Sparse matrices

Sparse and dense matrices

- Sparse matrices are sparsely populated by nonzero elements.
- Dense matrices have mostly nonzeros.
- Sparse matrix computations save time: operations with zeros can be skipped or simplified; only the nonzeros must be handled.
- Sparse matrix computations also save memory: only the nonzero elements need to be stored (together with their location).

Sparse matrix example

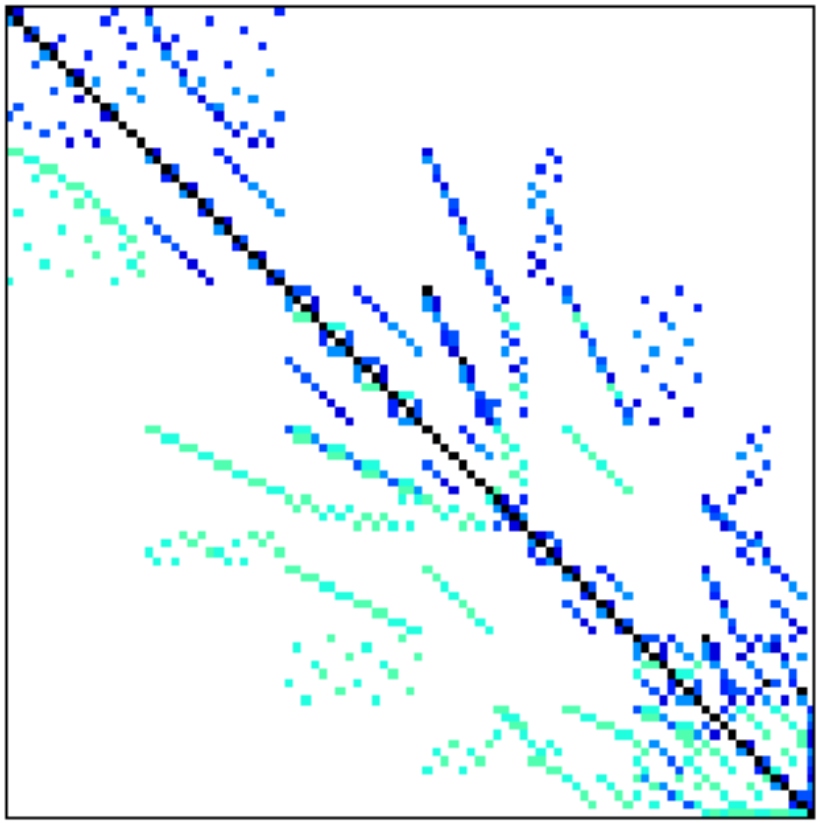


Figure 1: 93 rows and columns, 785 nonzeros, 8.4 nonzeros per row, 9.1% density.

Matrix statistics

- Number of nonzeros:

$$nz \equiv nz(A) \equiv |\{a_{ij} \mid 0 \leq i, j < n \text{ and } a_{ij} \neq 0\}|.$$

- Average number of nonzeros per row or column

$$c \equiv c(A) \equiv \frac{nz(A)}{n}.$$

- Nonzero density:

$$d \equiv d(A) \equiv \frac{nz(A)}{n^2}.$$

- A matrix sparse if $nz(A) \ll n^2$. Or, equivalently, when $c(A) \ll n$ or $d(A) \ll 1$.

Structure of sparse matrices

- If $a_{ij} \neq 0 \iff a_{ji} \neq 0$ then we say the matrix is structurally symmetric.
- This does not mean their values have to be equal. Computationally, the nonzero pattern is most important, not the the values.
- *Diagonal*, *tridiagonal* or more general banded matrices are also sparse.
- Sparse block matrices have a limited number of blocks, but these blocks can themselves be dense.

Irregular vs regular

- Regular algorithms have a computational cost that does not depend on the input. Examples of such algorithms you are familiar with are the FFT, LU, and dense matrix–matrix multiplication.
- Irregular algorithms, however, depend on the input. For sparse computations, they usually depend on the nonzero pattern of the matrix.
- Designing efficient irregular algorithms is a challenge. The ultimate goal is to make the algorithm as *efficient as possible for any input*.

Sequential algorithm

- An example of an irregular algorithm is the sparse matrix–vector product (SpMV).
- Given a sparse matrix A, and a dense vector v, compute $u \equiv Av$.

$$\boldsymbol{forall}\ (i,j)\ \mathit{such\ that}\ 0 \le i,j < n\ \mathit{and}\ a_{ij} \neq 0\ \boldsymbol{do}$$
$$u_i \leftarrow u_i + a_{ij} v_j$$

- The nonzero test $a_{ij} \neq 0$ is never executed in practice. Rather, a sparse data structure is used, or the nonzeros are generated on-the-fly.

Applications of SpMV

- Sparse matrices are the rule rather than the exception.
- In many applications, variables are connected to only a few others, leading to sparse matrices.
- Sparse matrices occur in various application areas:
  - transition matrices in Markov models;
  - finite-element matrices in engineering;
  - linear programming matrices in optimisation;
  - weblink matrices in Google PageRank computation.
  - molecular dynamics
- The sequential computation is simple, but its parallelisation is a big challenge.

Power method

- Power methods are based on repeated application of A to some initial vector. It finds the dominant eigenvector.
- Let A be a transition matrix, and $\vec{x}$ a vector of state frequencies (i.e., x_i is the relative frequency of state i).
- Computing $A\vec{x}, A^2\vec{x}, A^3\vec{x}, \ldots$ until convergence, we find a vector satisfying $A\vec{x} = \vec{x}$. This is the steady state.

Iterative methods

- More generally, sparse matrix—vector multiplication is the main computation step in iterative solution methods for linear systems or eigensystems.
- Iterative methods start with an initial guess x_0 and then successively improve the solution by finding better approximations x_k, $k = 1, 2, \ldots$, until the error is tolerable.
- Examples:
  - Linear systems $Ax = b$, solved by the conjugate gradient (CG) method or MINRES, GMRES, QMR, BiCG, Bi-CGSTAB, IDR, SOR, FOM, ...
  - Eigensystems $Ax = \lambda x$ solved by the Lanczos method, Jacobi–Davidson, ...

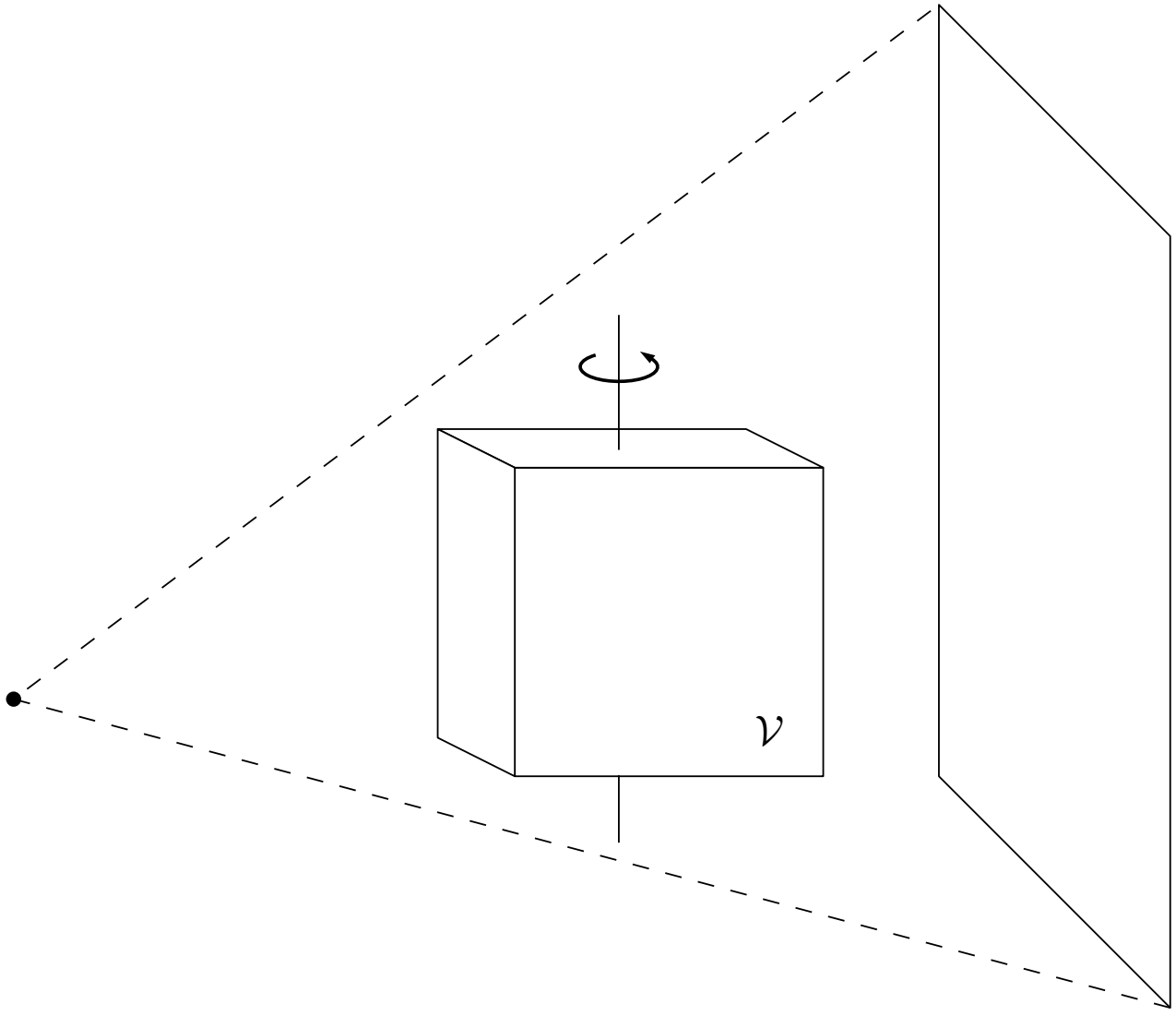
CWI

Tomography

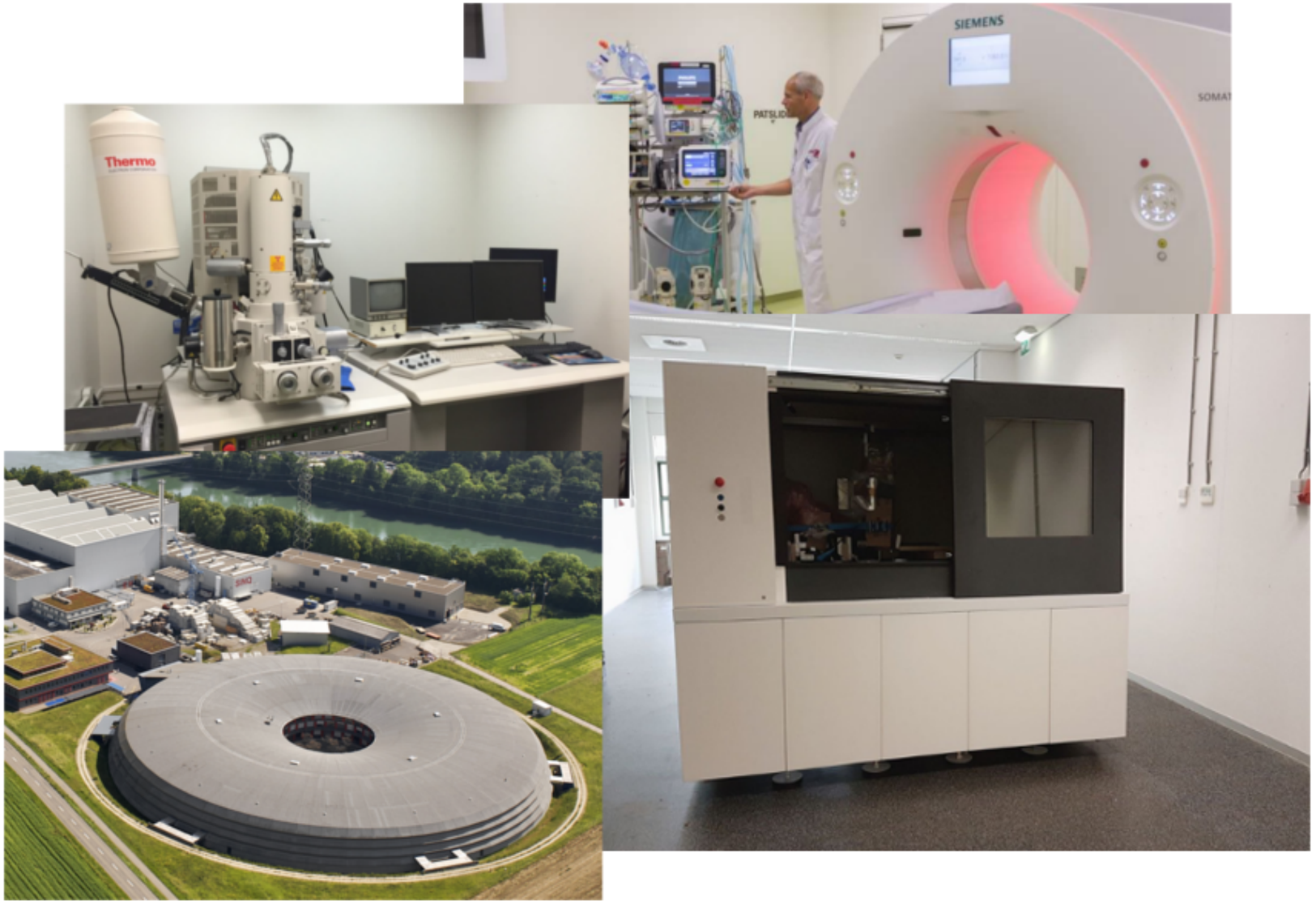
Introduction

- 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

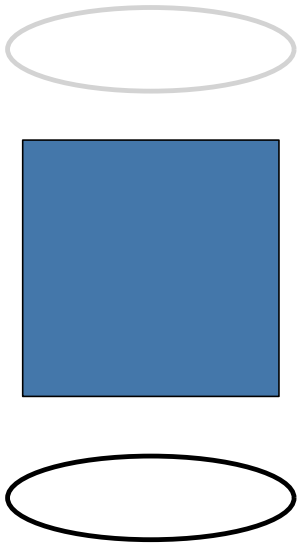
Example of tomographic measurement



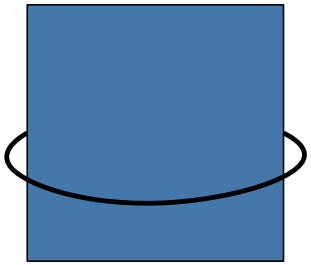
Applications of tomography



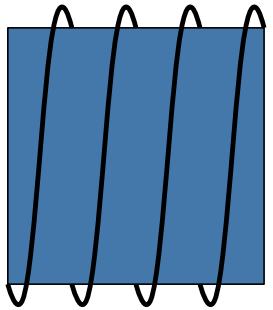
Acquisition geometries



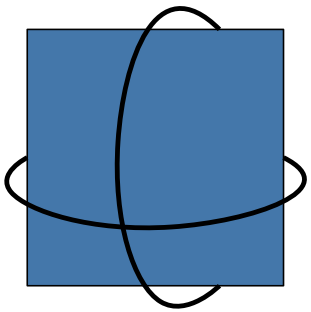
Laminography



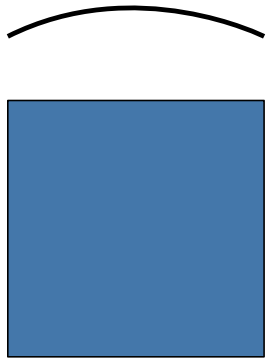
Single axis



Helical cone beam



Dual axis



Tomosynthesis

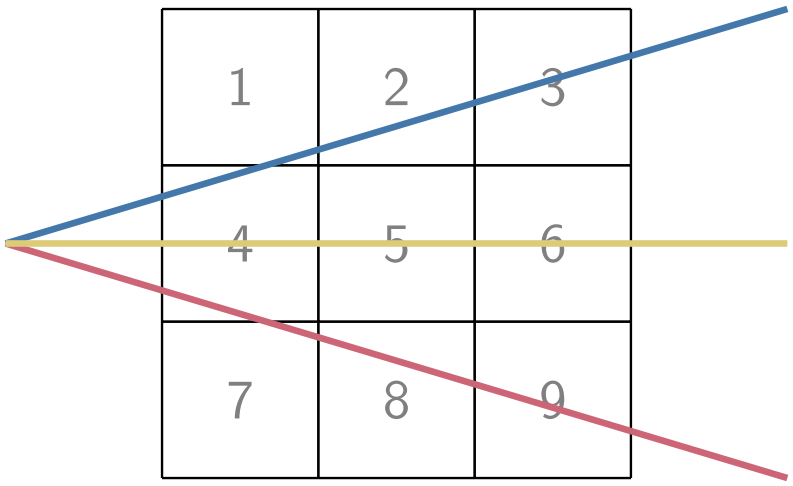
Tomographic reconstruction

- Projection matrix W, solve:

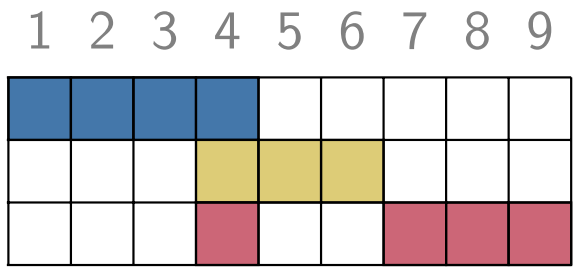
$$W\vec{x} = \vec{b},$$

  with $\vec{x}$ the image, and $\vec{b}$ the projection data.
- The projection data consists of a series of 2D images (the 'X-ray shadows' of the object), and are measured. The 3D image is unknown, and is to be reconstructed.
- 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

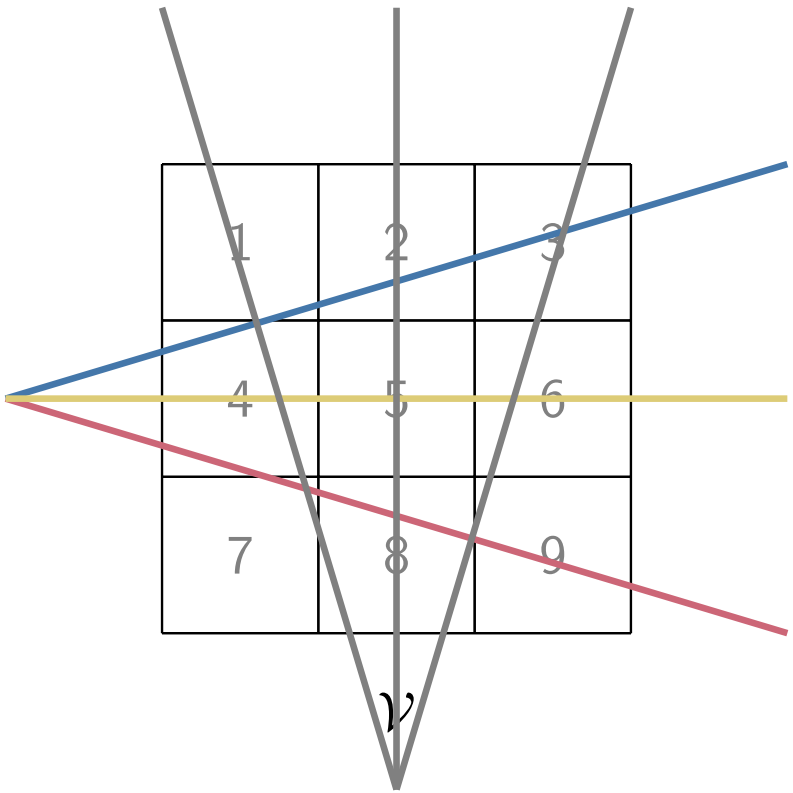


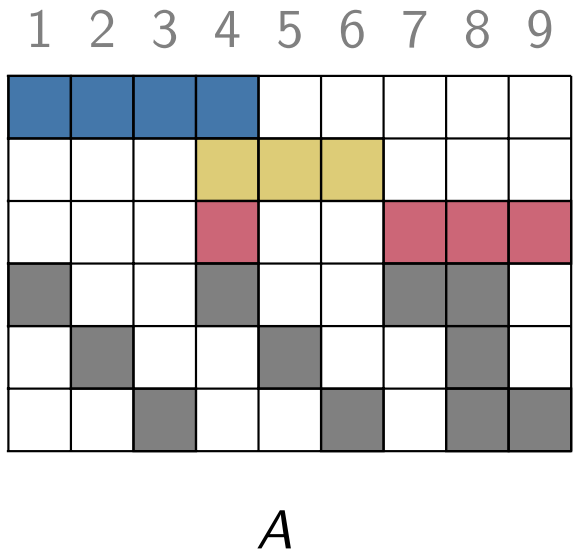
$\mathcal{V}$



A

Example of Projection Matrix (II)





Spy plot of a projection matrix

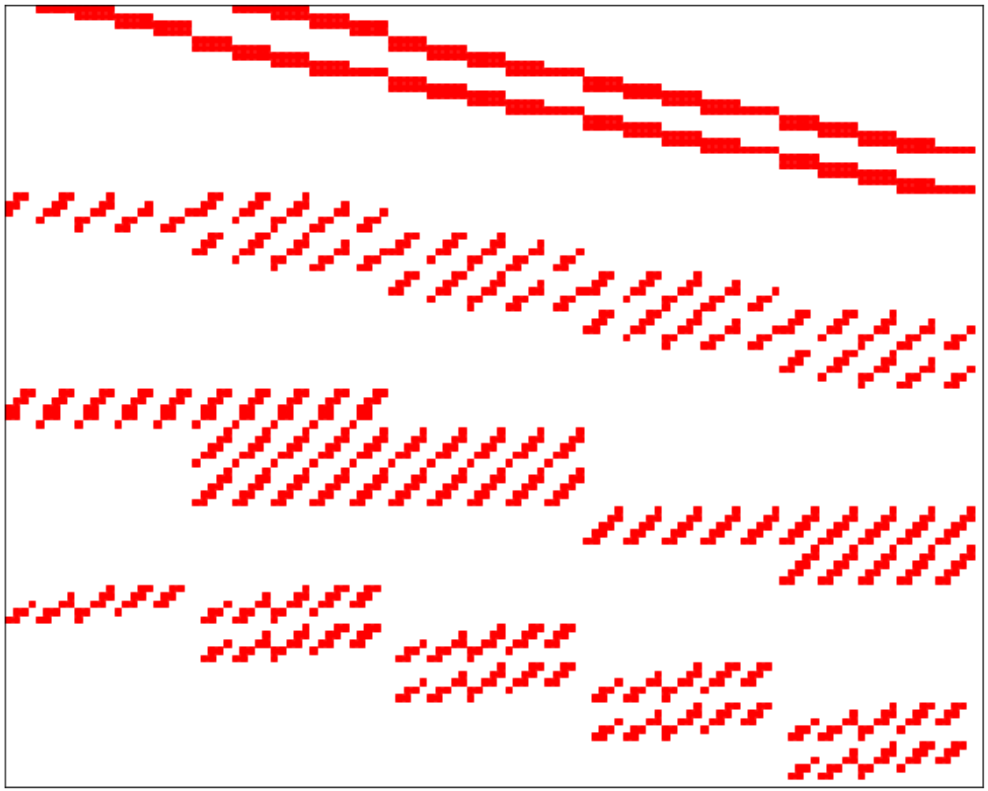


Figure 2: Parallel beam geometry matrix of size 100×125.

Large-scale tomography

- For tomographic reconstruction, the SpMVs $W\vec{x}$ and $W^T\vec{y}$ 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.

Distributed-memory tomographic image reconstruction

- In tomography, we are reconstructing (i.e. compute based on projection data) a 3D image.
- If we want to do this in parallel, we can make each processor responsible for reconstructing only a (small) part of the volume.
- However, rays cross the the entire volume, coupling these parts together. *How do we partition the image to minimize the coupling?*

Geometric partitioning problem

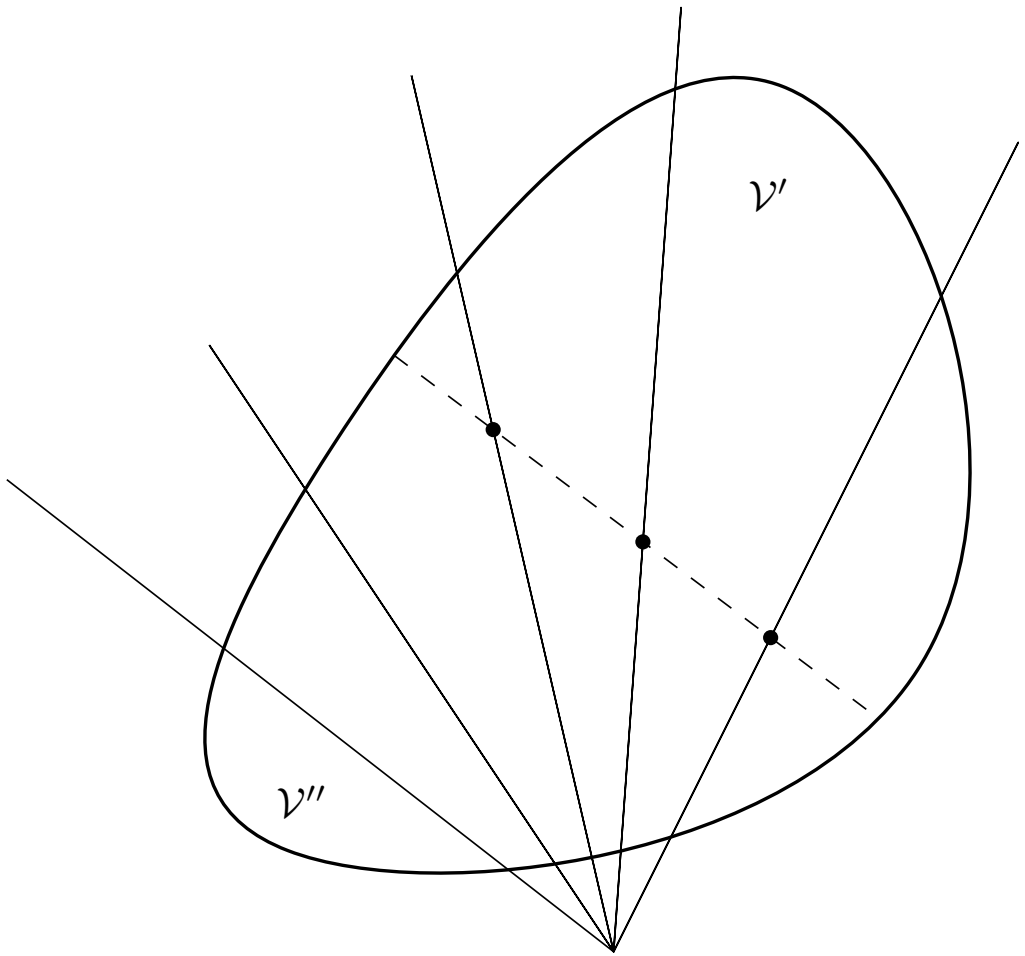
- We are given a cuboid, and a set of lines intersecting this cube.
- This cuboid is to be partitioned into p parts.
- A line crossing n parts has $n - 1$ cuts.
- What partitioning minimizes the total number of cuts?

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

- Communication volume equals number of lines through interface



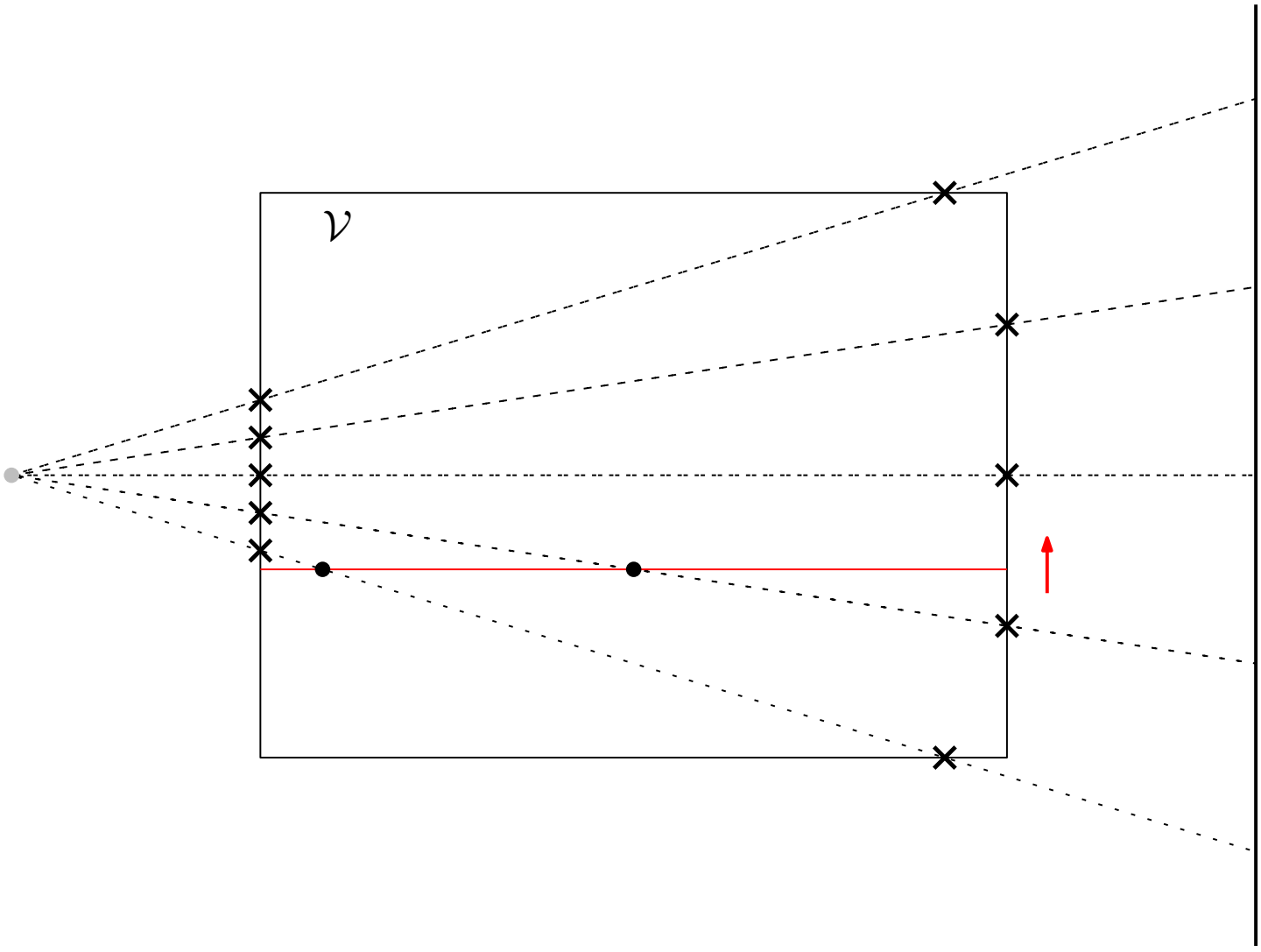
Bisectioning algorithm

- 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.

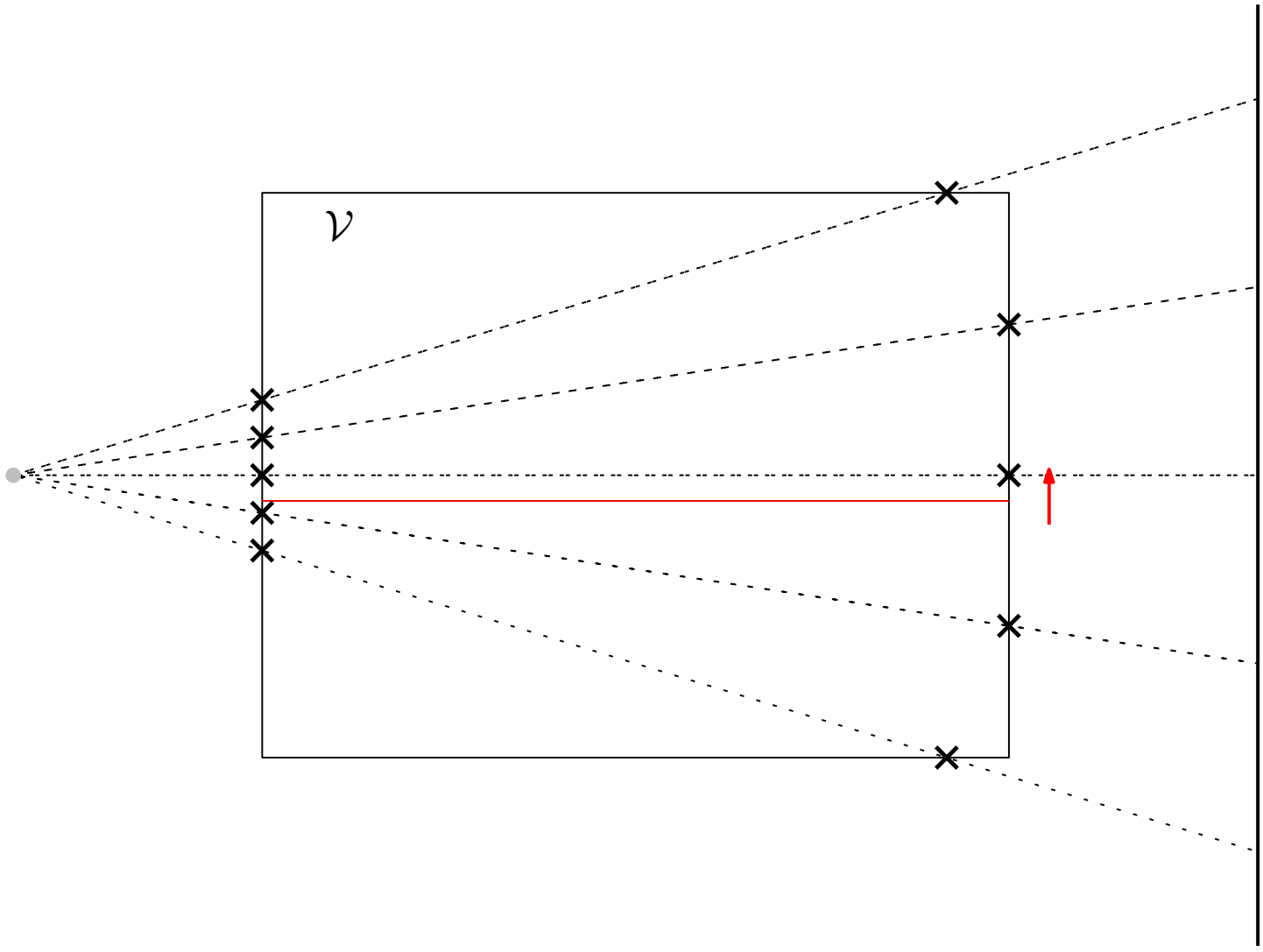
Plane sweep

- 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.

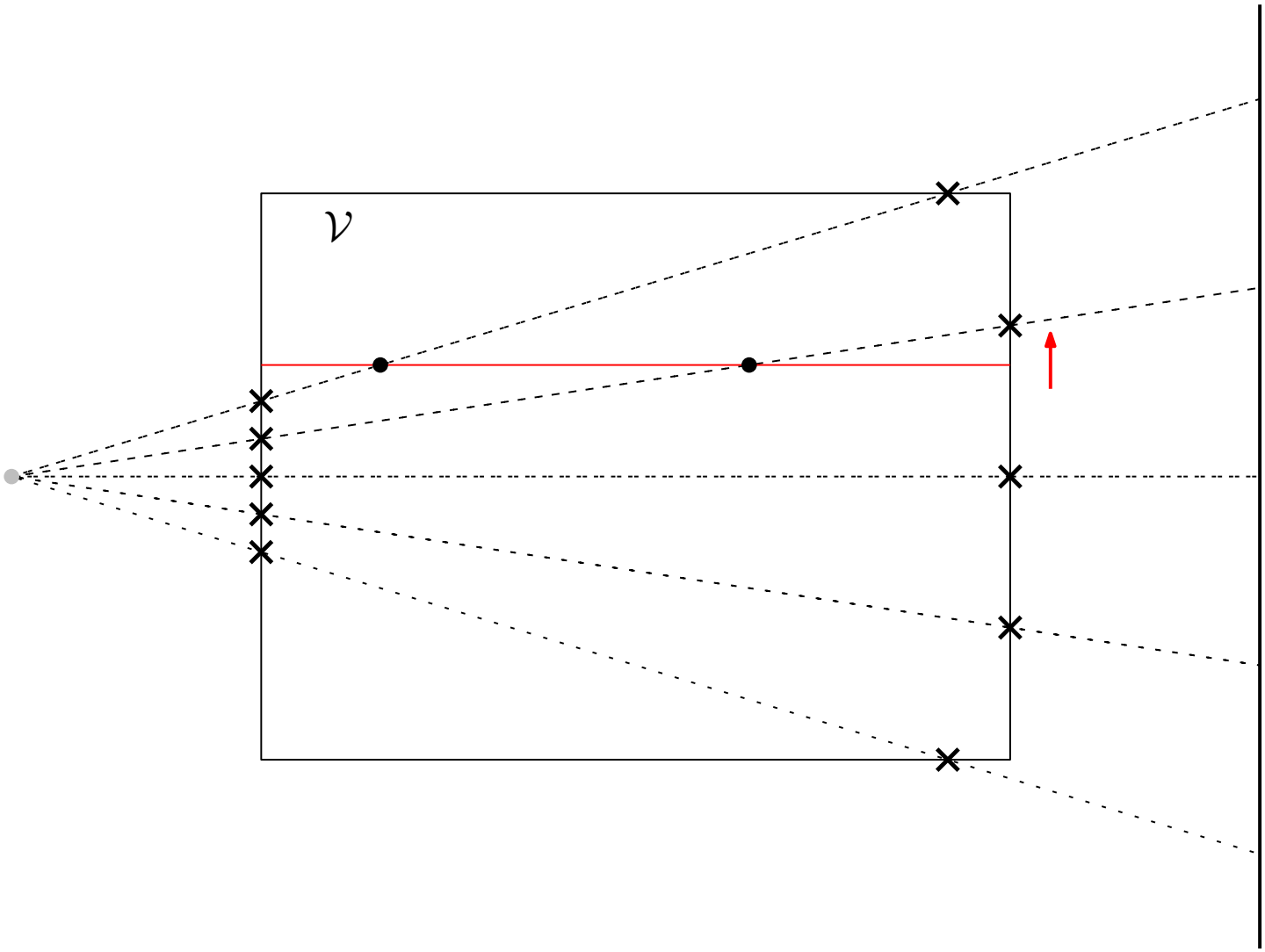
Example of plane sweep (I)



Example of plane sweep (II)



Example of plane sweep (III)

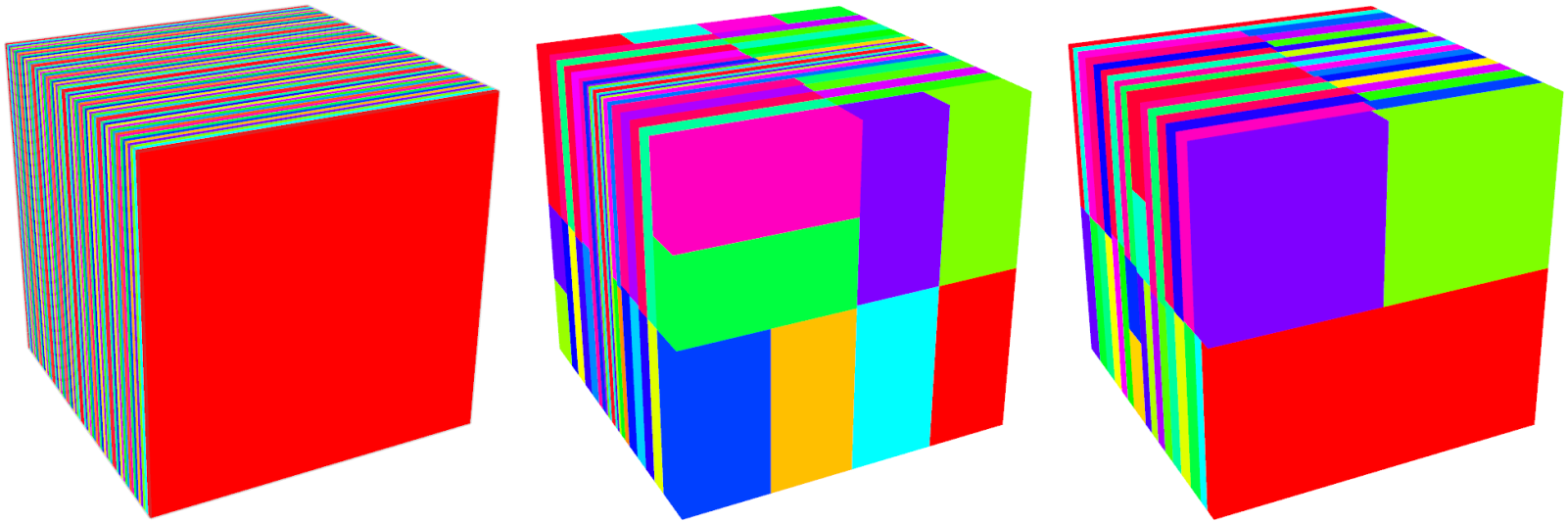


Results

- 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.

Partitioning results

- What constitutes a good partitioning depends heavily on the acquisition geometry.



- With a good partitioning, the amount of data that is communicated between processors is low.

Conclusion

- Sparse matrices are everywhere in scientific computing.
- Tomographic imaging is an important technique for science, medicine, cultural preservation and industry.
- Exploiting sparsity can save a lot of computational work. It requires, however, the design of specialized and irregular algorithms.
- Sequential sparse computations are relatively straightforward, but their parallelisation is a big challenge. Communication considerations often lead to interesting partitioning problems.

A geometric partitioning method for distributed tomographic reconstruction. Jan-Willem Buurlage, Rob Bisseling, Joost Batenburg. (Submitted to Parallel Computing)