Algorithms and methods for real-time tomography

Jan-Willem Buurlage, CWI Amsterdam

VORtech lunch talk, May 8th 2017

Introduction

Computational Imaging group at CWI

- Centrum Wiskunde & Informatica in Amsterdam.
- *Interdisciplinary* computational imaging group: physicists, mathematicians and computer scientists. Applications in *science, culture and industry.*
- Focus is on *advanced algorithms* and sofware for tomography
- We develop the ASTRA toolbox[1], many users around the world. Pioneers in GPGPU based tomographic reconstruction (way before my time in the group).

[1] http://astra-toolbox.com/, together with the University of Antwerp

Tomography

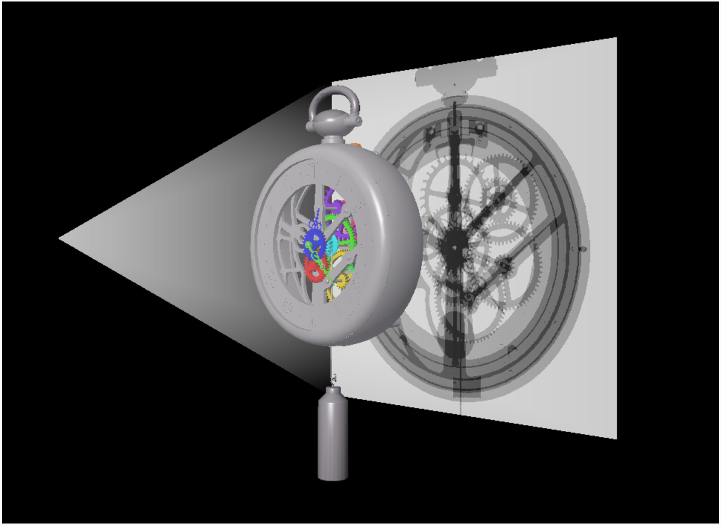


Figure 1: Typical tomography setup. X-rays are sent through a rotating sample, and projections are captured.

Algebraic algorithms

- For the attenuated rays, we numerically approximate integrals: object volume discretized into voxels.
- We get a projection matrix W, with a column for each voxel, a row for each ray.
- Tomographic reconstruction is now an inverse problem!

$$W\vec{x} = \vec{y}.$$

- Two special properties:
  - W is very large, too large to store explicitely.
  - Geometrical structure may get lost in the conversion.

Popular reconstruction algorithms

- FBP. Analytical method based on discretized inverse Radon transform.
- ART (Kaczmarz). Algebraic method, iterative scheme, satisfy each row in turn:

$$\vec{x}^{k+1} = \vec{x}^{k} + \frac{y_i - \langle \vec{a}_i, \vec{x}^k \rangle}{||a_i||^2} a_i^T .$$

- SIRT (Landweber). Simultaneous version of ART, weighted by the row- and column sums.

$$\vec{x}^{k+1} = \vec{x}^{k} + CW^T R(\vec{y} - W\vec{x}^k).$$

Real-time tomography

- Detectors are becoming larger, up to 4000×4000 pixels. Reconstruction volumes of 4000^3 voxels no longer unusual. 256 GB vectors!
- Time resolved experiments becoming more common. Multi-modal imaging, flexible acquisition geometries.
- We want real-time reconstructions as measurements come in, challenging computational problem.

Approaches for real-time tomographic reconstruction

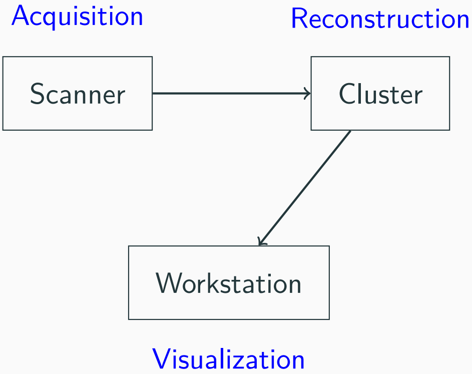
1. Write specialized software for real-time acquisition
2. Parallel and distributed algorithms
3. *Approximation algorithms (Rien)*

Software

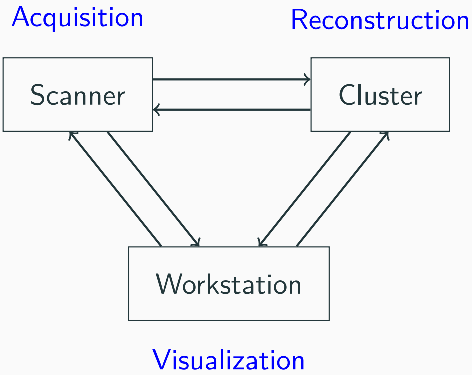
Software

- Idea: If it is infeasible to reconstruct the entire 3D volume, why not reconstruct individual slices? Analytical method can be used to reconstruct arbitrary slices.
- To create the illusion of having 3D reconstructions, we show the slices in context.
    - 2D slices together in 3D space.
    - Low resolution 3D preview.
- If it is easy to change the slices, then we have real-time quasi-3D reconstruction.

Simple pipeline



Complete pipeline



Slicing tool

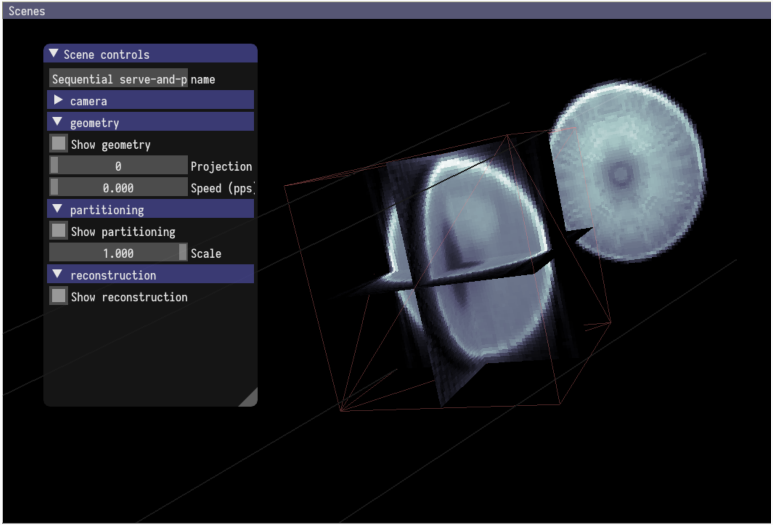


Figure 2: Proof-of-concept demo for real-time slicing tool

Technology

- Modern C++, few dependencies (ZeroMQ for communication, OpenGL for visualization), Python bindings for users
- Components:
  - Standardized description of acquisition geometries and data
  - Communication protocol using message passing
  - Visualization server as the control center

Distributed tomography

Approach II: Distributed methods

- Algebraic methods rely on the computation of $W\vec{x}$ (forward projection) and $W^T\vec{y}$ (back projection).
- Pseudo-code for e.g. $W\vec{x}$:

```
for (line : geometry) {
    for ([voxel, weight] : integrate(line)) {
        // [voxel, weight] represents W_ij
    }
}
```

Partitioning for distributed tomography

- Voxels assigned to processors
  - Can also distribute the geometry, but it is cheap to generate
- Each processor is responsible for computing projection values related to its voxels
- Intuition: when a ray goes through multiple subvolumes, those processors depend on each other's *results*.
- A new optimization problem: for a given geometry, find a good partitioning of the volume.

Partitioning

- Volume $\mathcal{V} \subset \mathbb{R}^3$. Acquisition geometry $\mathcal{G}$: set of lines through the volume.
- Partitioning π for p processors, s, t denote processors. Local volume: $\mathcal{V}^{(s)}$, such that $\bigcup_s \mathcal{V}^{(s)} = \mathcal{V}$.

$$\pi(\mathcal{V}) = \{\mathcal{V}^{(s)} \mid \forall_{s \neq t}\ \mathcal{V}^{(s)} \cap \mathcal{V}^{(t)} = \emptyset,\ 0 \leq s, t < p\}.$$

- Local geometry: $\mathcal{G}^{(s)}$: all lines that cross the local volume.

Partitioning problem

- Line weight

$$\alpha(\ell) = |\{s \mid \ell \in \mathcal{G}^{(s)}\}|, \ \alpha(\mathcal{G}) = \sum_{\ell \in \mathcal{G}} (\alpha(\ell) - 1).$$

- Voxel weight

$$\beta(x_j) = |\{\ell \in \mathcal{G} \mid x_j \in \ell\}|, \ T^{(s)} = \sum_{x_k \in \mathcal{V}^{(s)}} \beta(x_k).$$

$$\eta(\pi) = \max_{0 \leq s < p} \frac{T^{(s)}}{T_{\text{avg}}}.$$

- Partitioning problem:

$$\begin{aligned} \text{minimize}_{\pi} \quad & \alpha(\mathcal{G})(\pi) \\ \text{subject to} \quad & \eta(\pi) < \eta_{\max}. \end{aligned}$$

Partitioning methods

- The importance of good partitionings for parallel SpMV's are well known in the sparse matrix community.
- Many software packages and methods tackle this problem. None of them applicable to tomography, W is implicit.
- Need to make use the geometrical nature of the problem.

Geometric partitioning

- Problem summary: for a set of lines $\mathcal{G}$, equally partition a volume $\mathcal{V}$ so that overall, the lines cross as few different subvolumes as possible.

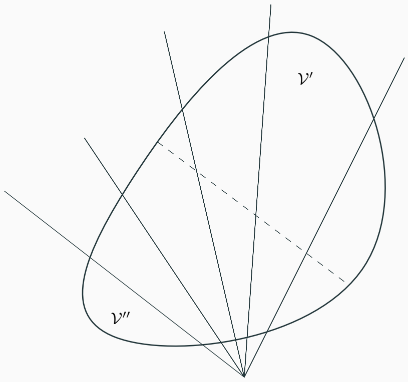


Figure 3: Communication is # interface intersection points.

Geometric partitioning (II)

- Recursive bisectioning: split the volume into two subvolumes recursively.

Theorem

Let $\mathcal{V} = \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_n$ be a cuboid partitioning. Then for any acquisition geometry $\mathcal{G}$ we have:

$$\begin{aligned}\alpha(\mathcal{G})(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n) = \alpha(\mathcal{G})(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_{n-1} \cup \mathcal{V}_n) \\ + \alpha(\mathcal{G})(\mathcal{V}_{n-1}, \mathcal{V}_n).\end{aligned}$$

- Conclusion: recursively bisecting is warranted, only worry about bisecting.

Plane sweep

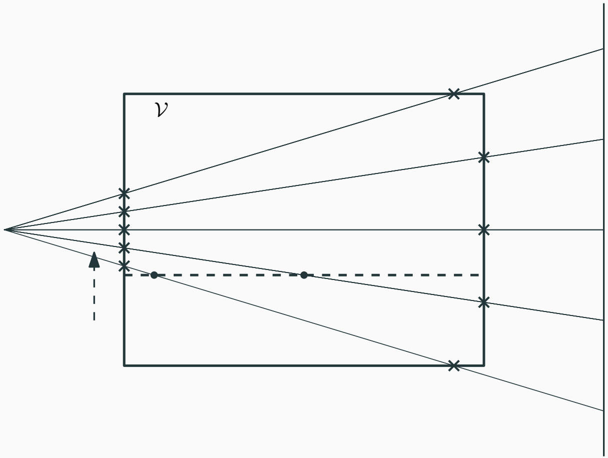


Figure 4: When splitting in two, we can use a plane sweep algorithm.

Distributed tomography software

- Parallel code based on the BSP model, the basis for MapReduce and Pregel.
- Mathematical model for reasoning about parallel algorithms.
- Open-source software: Bulk[2], Epiphany BSP[3].
- For tomography, Tomos[4] toolbox.

---

[2] http://www.github.com/jwbuurlage/bulk/
[3] http://www.github.com/coduin/epiphany-bsp/
[4] http://www.github.com/jwbuurlage/tomos

Summary

- Generic real-time tomography only possible when combining software, parallel algorithms, and mathematics.
- Specialized software can be of great help to experimenters and algorithm designers in tomography.
- Efficiency requirements for distributed algorithms can give rise to rich optimization problems for which new algorithms have to be developed.