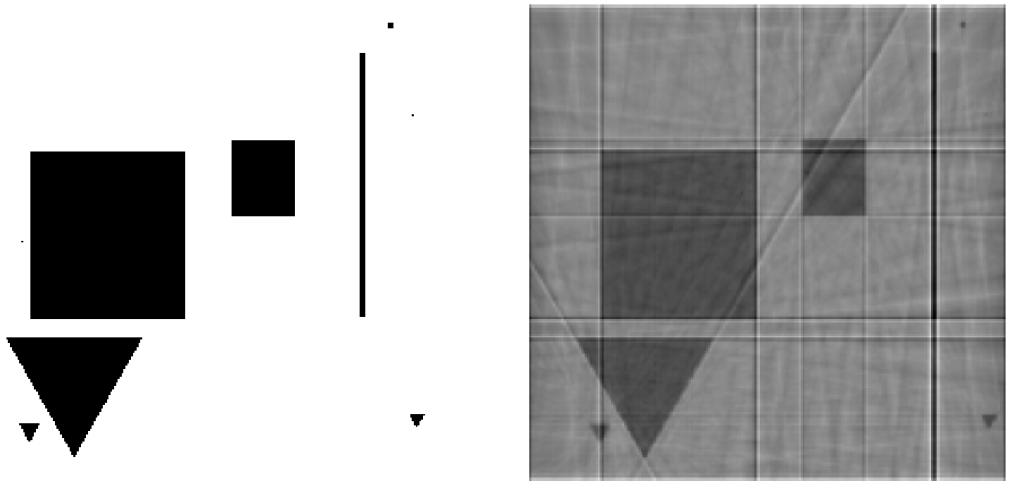
Using Futhark for SIRT

Mette Bjerg Lindhøj

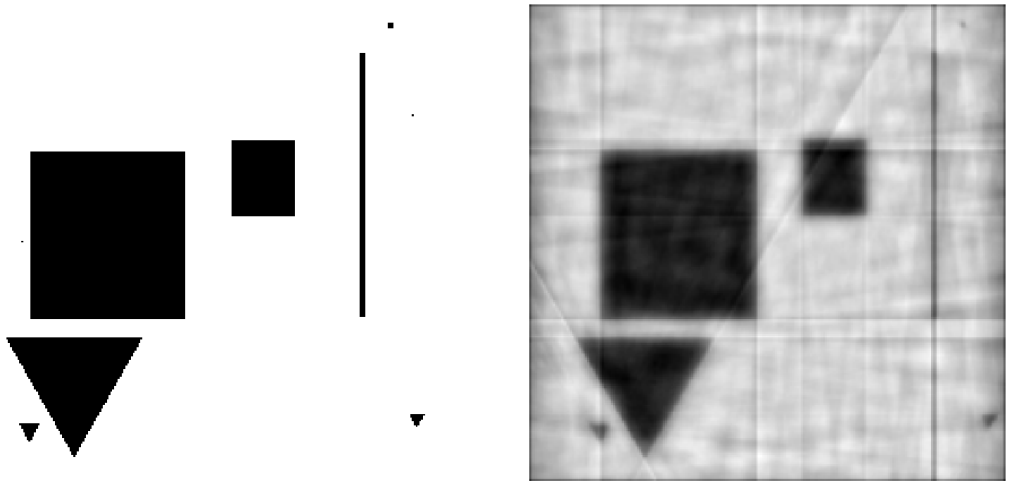
University of Copenhagen

17/12/2018

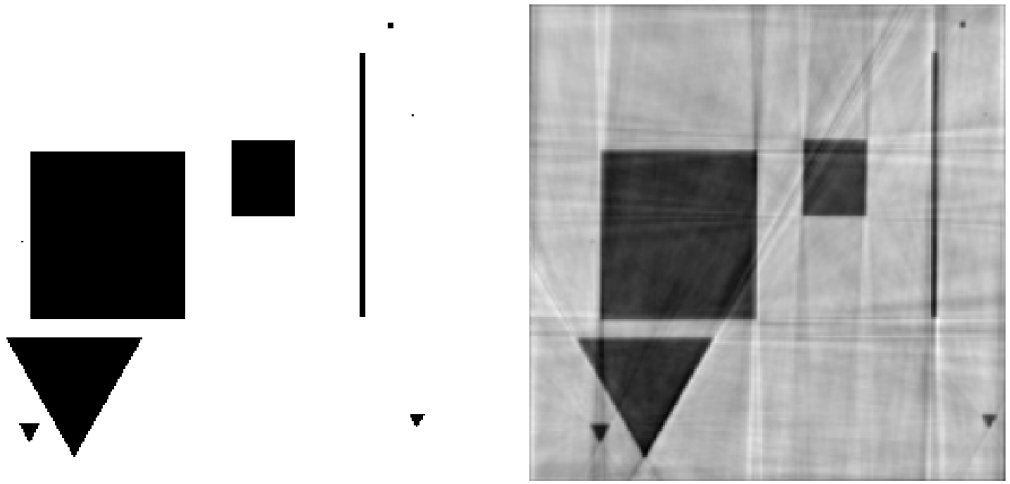
MSE: 0.21, SSIM: 0.60



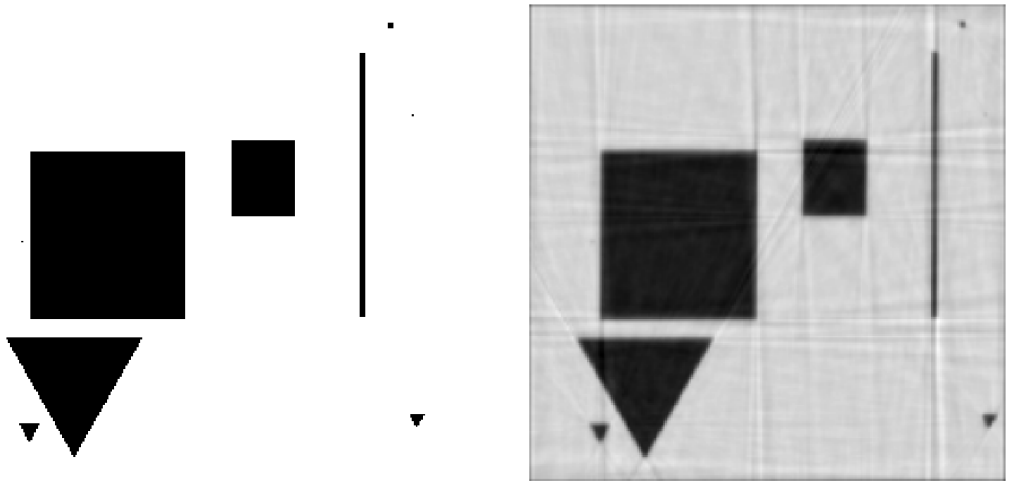
MSE: 0.08, SSIM: 0.69

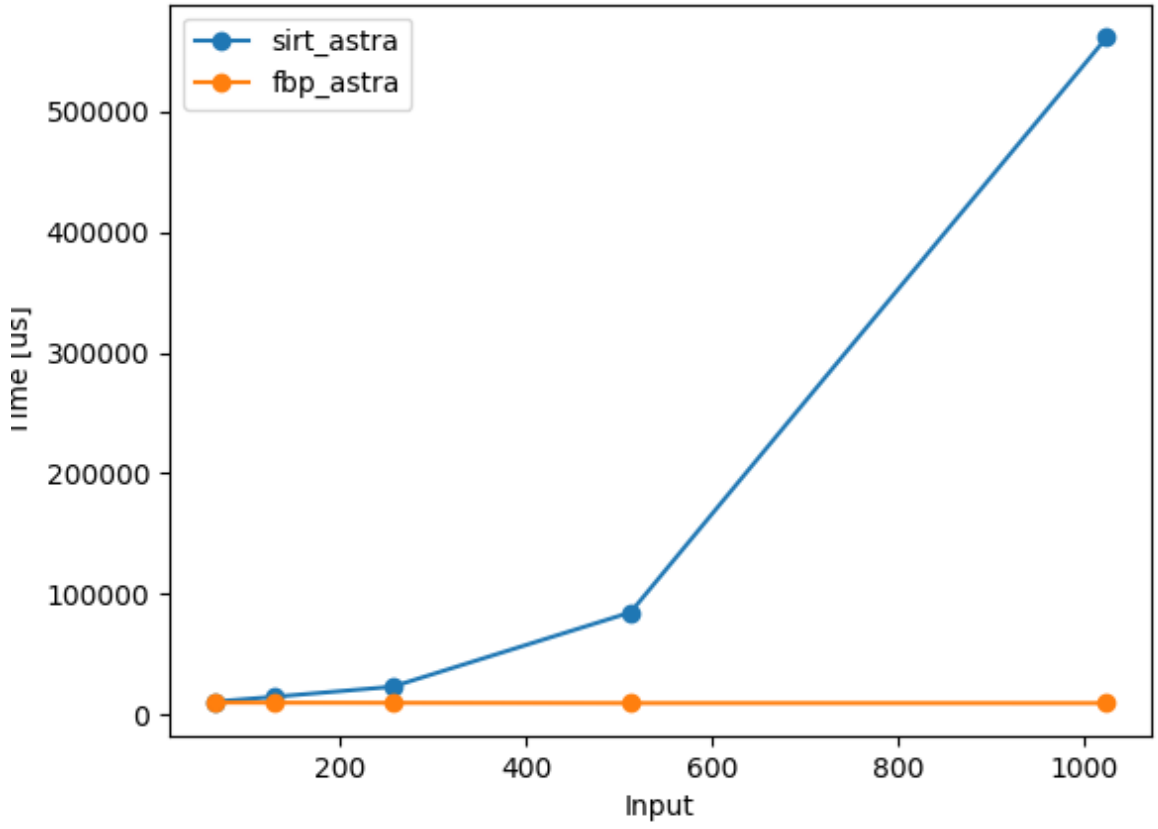


MSE: 0.09, SSIM: 0.66



MSE: 0.05, SSIM: 0.73




sirt_astra
fbp_astra
500000
400000
300000
200000
100000
0
Time [us]
200
400
600
800
1000
Input

SIRT

Solve the problem:

$$\boldsymbol{f}^* = argmin_{\boldsymbol{f}} \|\boldsymbol{p} - \boldsymbol{A}\boldsymbol{f}\| \tag{1}$$

iteratively using this update step:

$$\boldsymbol{f}^n = \boldsymbol{f}^{(n-1)} + \boldsymbol{C}\boldsymbol{A}^T\boldsymbol{R}(\boldsymbol{p} - \boldsymbol{A}\boldsymbol{f}^{(n-1)}), \tag{2}$$

where $\boldsymbol{C}$ and $\boldsymbol{R}$ are the diagonal matrices containing the inverse column and row sums of the system matrix respectively.

When is something parallel?

Consider a loop. This loop is parallel if there are no two iterations of the loop i and j, where $i < j$ such that the computations of iteration j:

RAW read data from a memory location written to during iteration i

WAR write a value to a memory location read during iteration i

WAW write data to a memory location written to during iteration i

Simply put: The computation of each iteration is independent of the computations performed during previous iterations.

Why implement our own in Futhark?

- Hardware agnostic
- Highlevel language
- MemoryError

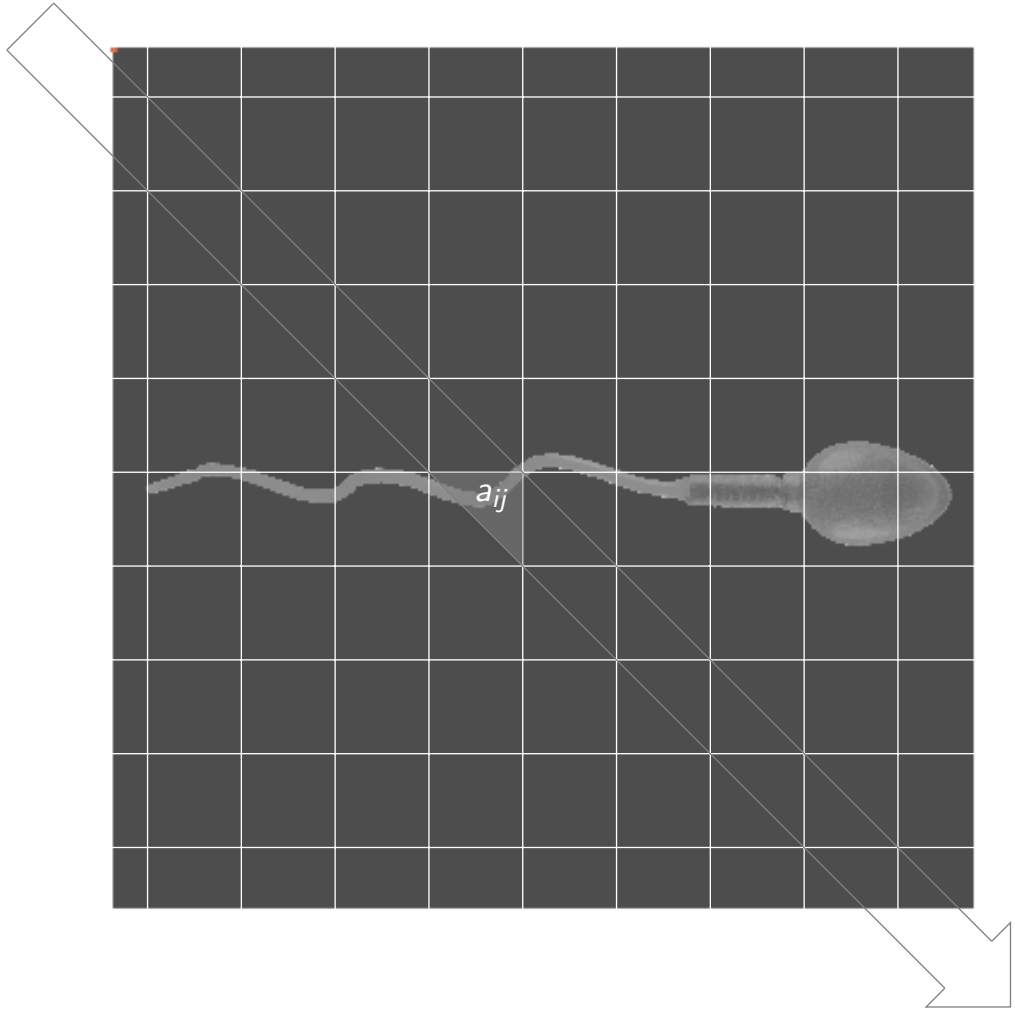
```
% Input: sparse system matrix A, data b.
% Output: SIRT reconstruction x.
x = zeros(d * d, 1);
[rows cols] = size(A);
C = sparse(1 : cols, 1 : cols, 1 ./ sum(A));
R = sparse(1 : rows, 1 : rows, 1 ./ sum(A'));
CATR = C * A' * R;
for i = 1 : 100
  x = x + CATR * (b - A * x);
end
```

Figure: Example of high level implementation

```
1  unsigned int blockStart = 0;
2  unsigned int blockEnd = 0;
3  bool blockVertical = false;
4  for (unsigned int a = 0; a <= dims.iProjAngles; ++
       a) {
5    bool vertical = false;
6    if (a != dims.iProjAngles)
7      vertical = (fabsf(angles[a].fRayX) <= fabsf(
           angles[a].fRayY));
8    if (a == dims.iProjAngles || vertical !=
         blockVertical) {
9
10     blockEnd = a;
11     if (blockStart != blockEnd) {
12       dim3 dimGrid((blockEnd-blockStart+
             g_anglesPerBlock-1)/g_anglesPerBlock,
13                   (dims.iProjDets+g_detBlockSize
                       -1)/g_detBlockSize); //
                       angle blocks, detector
                       blocks...
```

Figure: Example of CUDA implementation

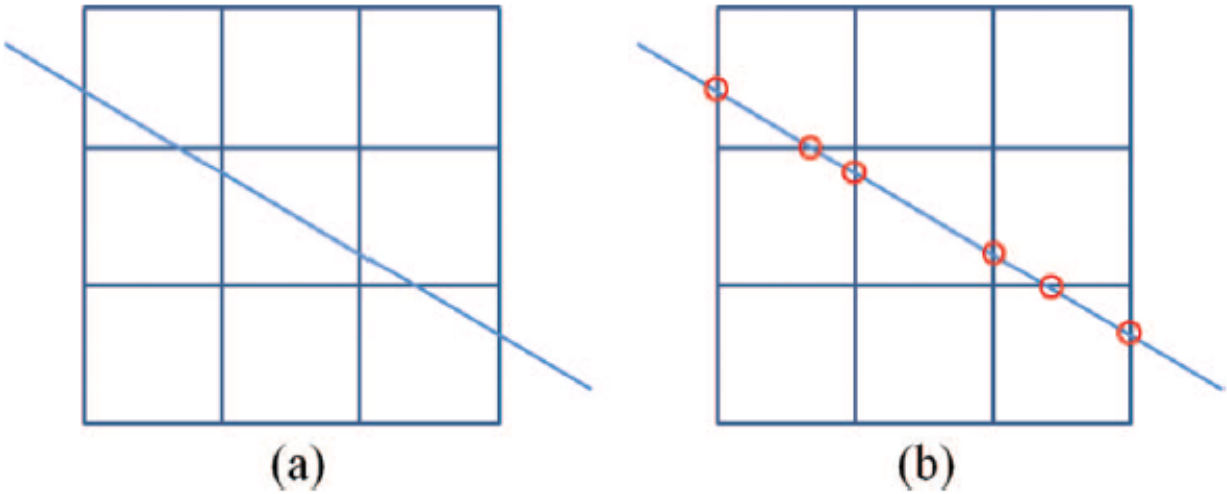
The system matrix

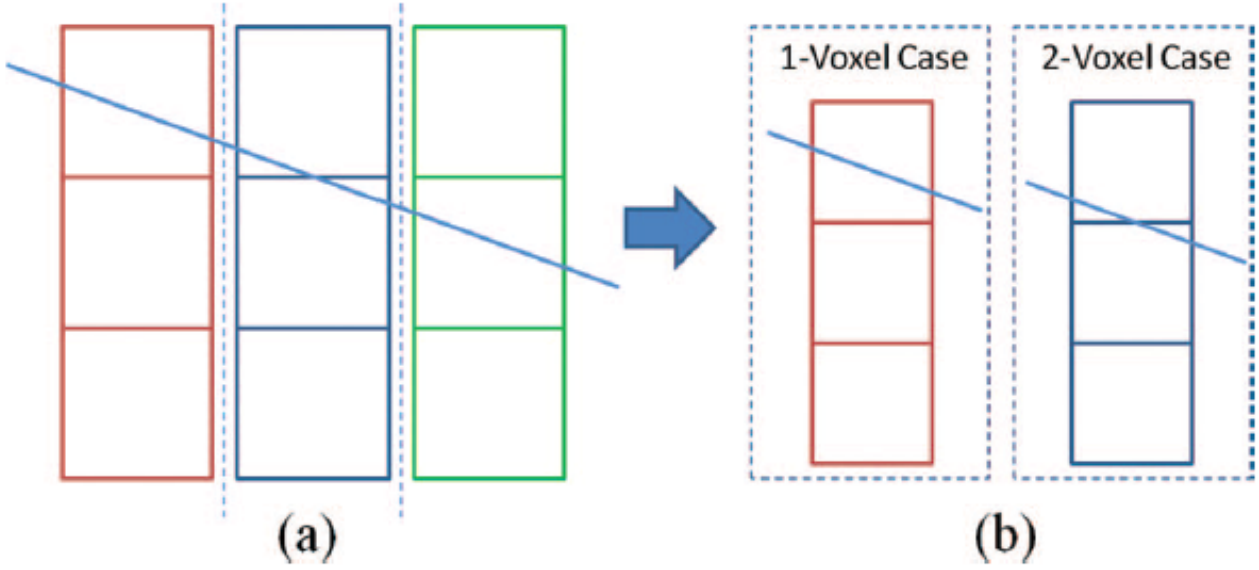


The problem is in the size

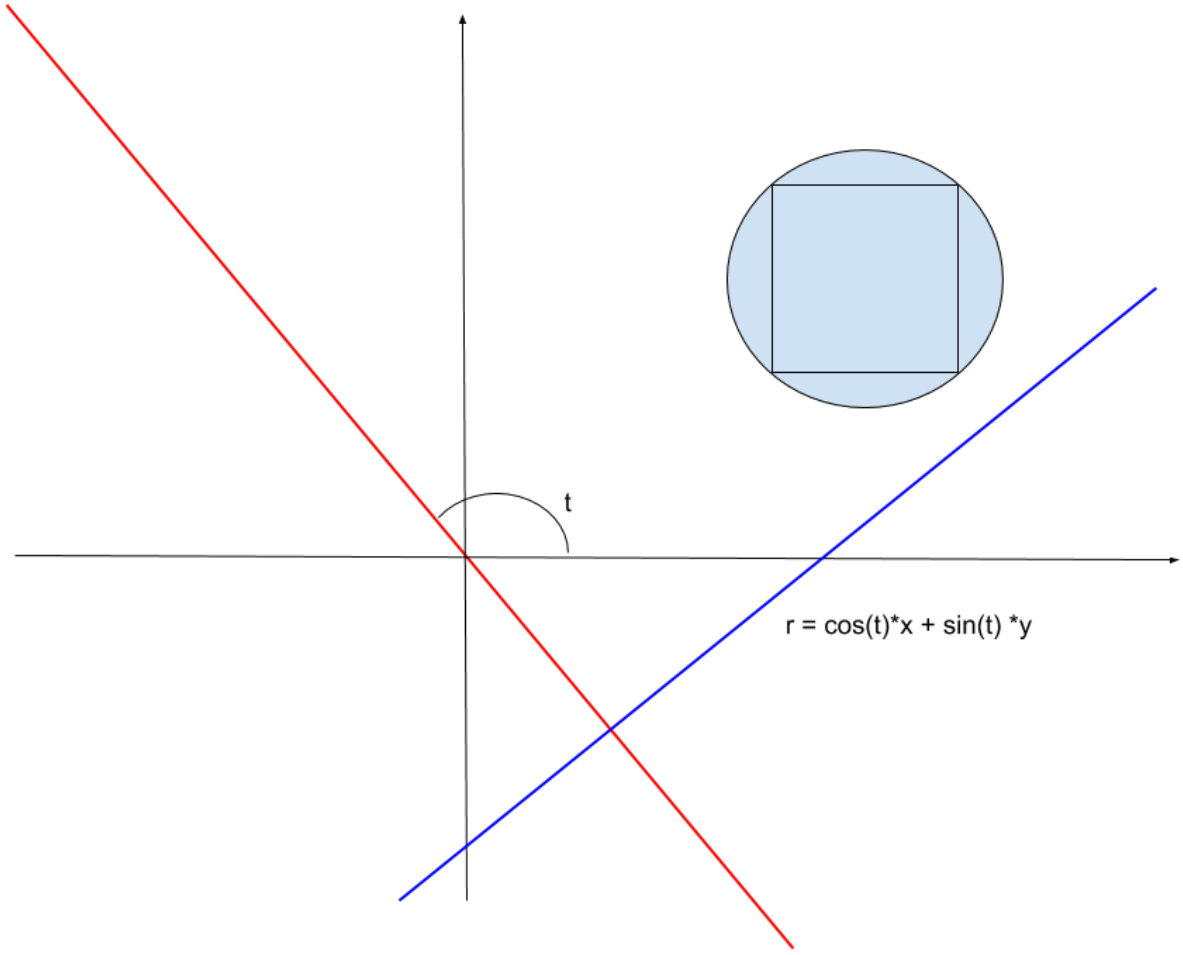
- Consider reconstructing a single slice of a volume from a detector of size $n \times n$
- The number of rays is n
- The number of angles is $\frac{n \cdot \pi}{2}$
- A typical value for n is 2048
- In semi sparse format the matrix will take up $2 \cdot 4 \cdot 2048 \cdot \lceil \frac{2048 \cdot \pi}{2} \rceil \cdot (2 \cdot 2048 - 1) \approx 216 GB$

System matrix computation

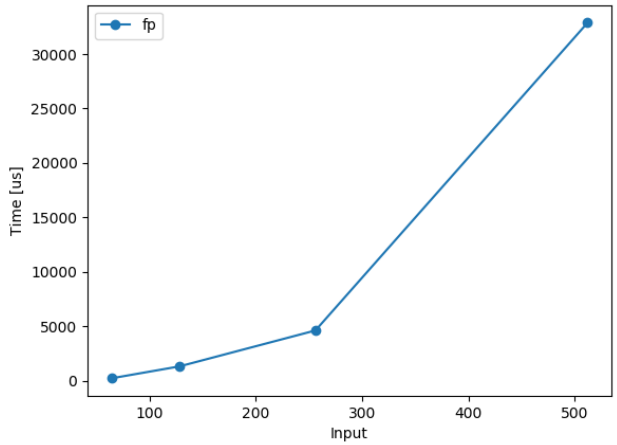


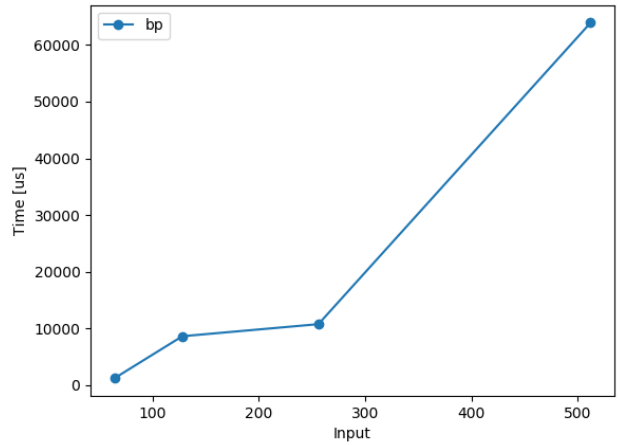


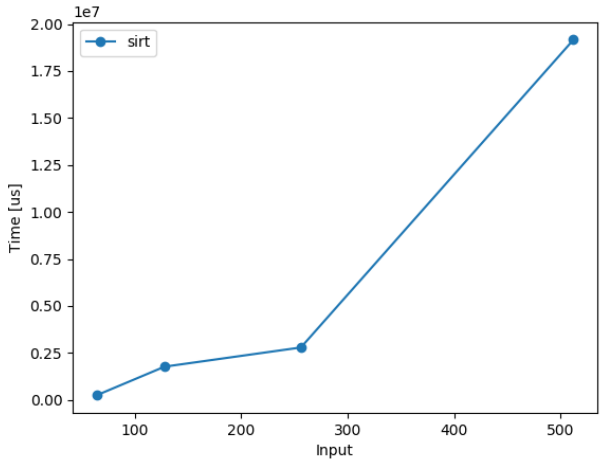
Transposed system matrix computation



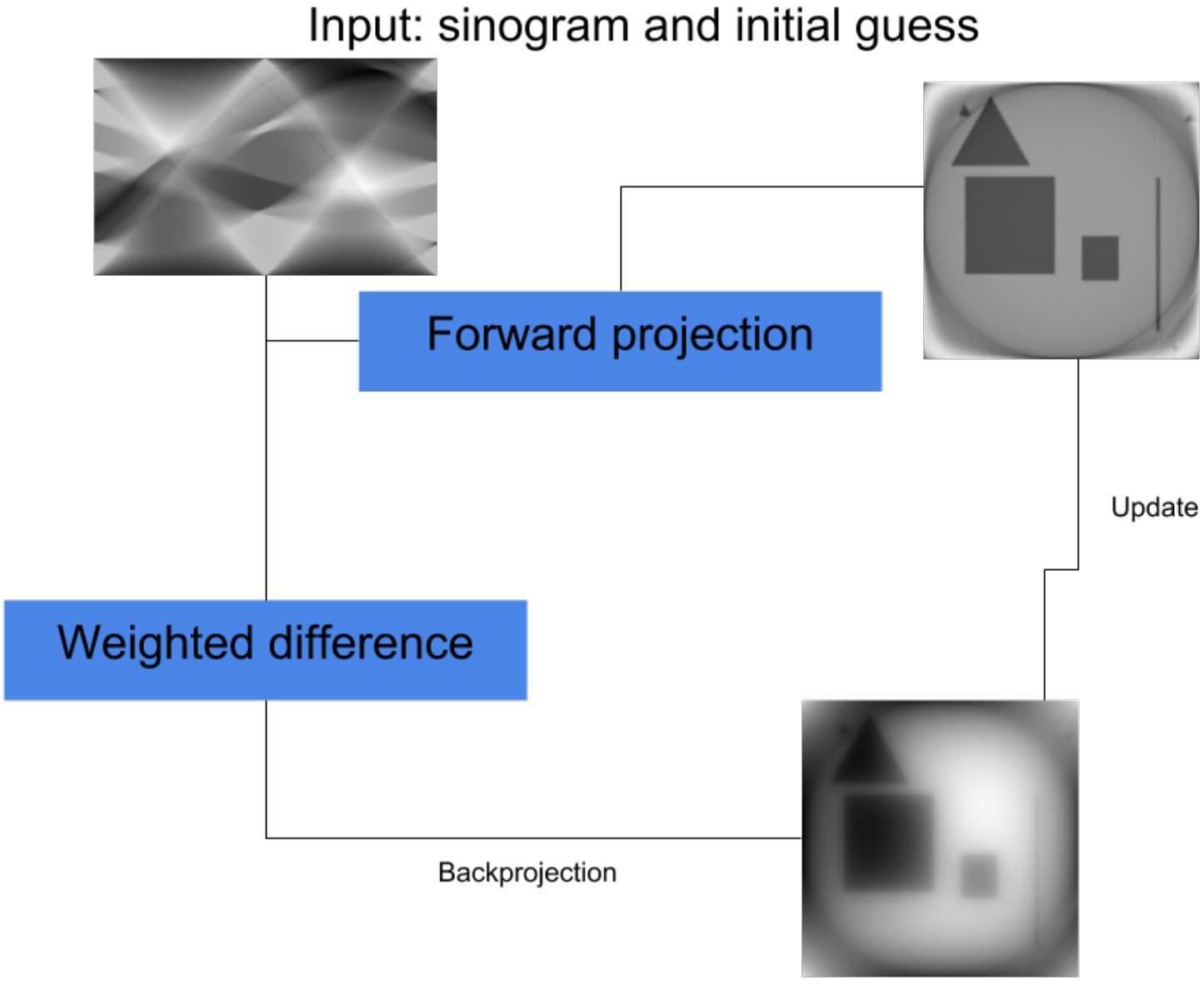
Status


fp
30000
25000
20000
15000
10000
5000
0
Time [us]
100
200
300
400
500
Input


bp
60000
50000
40000
30000
20000
10000
0
Time [us]
100
200
300
400
500
Input


1e7
sirt
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Time [us]
100
200
300
400
500
Input

Status


Input: sinogram and initial guess
Forward projection
Update
Weighted difference
Backprojection

Future plan

- Fix the bug in backprojection
- Ask our expert for obvious improvements
- Bachelor project aimed at solving memory issues during the spring
- Optimizing matrix computation based on symmetries
- Implementing for cone beams
- Extending to 3D
- Running on the image cluster and comparing to ASTRA toolbox
- Use as part of joint reconstruction and motion estimation algorithm


Hao Gao.
Fast parallel algorithms for the x-ray transform and its adjoint.
Medical physics, 39(23127102):7110–7120, November 2012.


Y. Long, J. A. Fessler, and J. M. Balter.
3d forward and back-projection for x-ray ct using separable footprints.
IEEE Transactions on Medical Imaging, 29(11):1839–1850, Nov 2010.


F. Natterer.
The Mathematics of Computerized Tomography.
Society for Industrial and Applied Mathematics, 2001.


Z. Xue, L. Zhang, and J. Pan.
A new algorithm for calculating the radiological path in ct image reconstruction.
In *Proceedings of 2011 International Conference on Electronic Mechanical Engineering and Information Technology*, volume 9, pages 4527–4530, Aug 2011.