Subtomogram Averaging: Workflows and Pipelines

How-to guide

Daniel Castaño-Díez
Scientific Computing Unit
Max-Planck Institute for Brain Research, Frankfurt
Contents

• Introduction
  - Structure determination by Subtomogram Averaging

• Pipelines in Subtomogram averaging:
  - Setting up a refinement experiment
  - Classification.
  - Extracting particles
  - Using High Performance Computing
Architecture of the desmosomal junction: Identification of individual cadherines

Subtomogram Averaging

raw data: many noisy copies of a protein

averaged 3d model
Tomography pipeline with **subtomogram averaging** (structure determination of macromolecules)
Core operations:

- volume rotation
- FFT for cross correlation
... and the world is full of incarnations of this idea...
How standards proliferate:
(See: A/C chargers, character encodings, instant messaging, etc.)

**Situation:**
There are 14 competing standards.

14?! Ridiculous!
We need to develop one universal standard that covers everyone's use cases.

Yeah!

**Soon:**

**Situation:**
There are 15 competing standards.
The Subtomogram Averaging Procedure

unsolved questions:
* template bias
* missing wedge
* CTF correction
* sample thickness
* resolution measurement
* heterogeneity

does not exist

so, what do we ideally ask from a software package for subtomogram averaging?

NO (reliable)
BLACK BOX

Repetition and interactive analysis of experiments

exhaustive computations on volumetric data

exhaustive computations on volumetric data

very numerically demanding

user friendliness

user flexibility

high numerical performance

High Performance Computing

ARTIFACTS
The ingredients of Subtomogram Averaging

1. **Data**
2. **Mathematics**
3. **Hardware**
4. **Visualization**
Data

Set of individual files.
Entities in a database.

whichever format you use:

Did the cropping procedure work as expected?
Is the dynamical range correct?

inspecting your particles:
always a good idea
Metadata

At the particle level:
• Initial orientation (euler angles)
• Initial shift
• Position in tomogram

Many formats:
• Dynamo table, jsubtomo star file, Xmipp doc…..
• Conversion tools available.

Check compatibility with data!

rough orientations
Initial Reference

Avoid using high resolution structures

*A wrong iterative procedure may easily create a “template bias” effect.*
References: averages of (subsets of ) data particles

*Use coarse orientations if known.*
*Otherwise use random orientations/symmetrization to avoid missing wedge in the template.*
Bacterial type III secretion system in *Yersinia* bacterium

Initial reference by manual alignment
avoid elements that are not part of the signal

Systematically:
• membranes
• flexible parts

On individual particles:
• Gold beads
• Crowded environment

SIZE TRADE OFF

Include enough signal to drive the alignment
**Masks**

Alignment procedure should ignore the non-coherent signal of each particle.

1) Use default masks provided by most programs (sphere, ellipsoid).

2) Use combinations of geometrical shapes.

- Systematically:
  - membranes
  - flexible parts

- On individual particles:
  - Gold beads
  - Crowded environment

Defines the part in the 3d template that is actually compared to the contents of each data particle.
... use a hand-drawing tool to precisely define the mask on the domain interest...

Normally manual input will only be needed for some slices. Shape of the handdrawn mask in intermediate slices can be interpolated.

- avoid elements that are not part of the signal

SIZE TRADE OFF

Include enough signal to drive the alignment
Designing the iteration process

<table>
<thead>
<tr>
<th>Parameters</th>
<th>round 1</th>
<th>round 2</th>
<th>rc</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterations</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>references</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>cone aperture</td>
<td>360</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>cone sampling</td>
<td>60</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>cone flip</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>check orientation peak</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>freeze reference</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>azimuth rotation range</td>
<td>0</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>azimuth rotation sampling</td>
<td>1</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>azimuth flip</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>check azimuth peak</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>freeze azimuth reference</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>high pass</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>low</td>
<td>21</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>symmetry</td>
<td>c8</td>
<td>c8</td>
<td>c8</td>
</tr>
<tr>
<td>Particle dimensions.</td>
<td>32</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>refine</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>refine factor</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>shift limits</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1</td>
</tr>
<tr>
<td>shift limiting way</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>separation in tomogram</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>check peak in xy shift</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>check peak in z shift</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fourier constraints</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
.... you can let golden standard based –adaptive filtering drive.....

.... Slightly adapted to different geometries.....
Designing the iteration process

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Round 1</th>
<th>Round 2</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>References</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cone aperture</td>
<td>360</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Cone sampling</td>
<td>60</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>Check orientation peak</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Freeze reference</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Azimuth rotation range</td>
<td>0</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>Azimuth rotation sampling</td>
<td>1</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Azimuth flip</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Check azimuth peak</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Freeze azimuth reference</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>High pass</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Low</td>
<td>21</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Symmetry</td>
<td>c1</td>
<td>c8</td>
<td>c8</td>
</tr>
<tr>
<td>Particle dimensions</td>
<td>32</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>Refine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refine factor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift limits</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Shift limiting way</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Separation in tomogram</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Check peak in xy shift</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check peak in z shift</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fourier constraints</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Estimate total runtime!

Multigrid search:

A cheap alternative to global angular scan
The effect of multigrid angular sampling

data:
Synthetic thermosome template
64x64x64 voxels
120 copies
randomly oriented
missing wedge
Low noise.

Let’s run two alignment projects with the same parameters except angular sampling
The effect of angular sampling

**Approach 1**
1 iteration
app. 200 angles
global sampling

**Approach 2**
1 iteration app.
app 200 angles
regular sampling
+ local refining
Shift limits

Cross correlation peak giving the right shift is well defined and a global maximum.
Real data is more complicated....
refinement and average without shift limits

same refinement parameters + shift limits
3 Hardware

...and go for it.....

choose environment

... and resources.....
...and you are done...
... and you are done...

... well,
you might need
some auxiliary tools
Classification

Many currently used approaches are variants of one of these two main flavors:

**PCA + Kmeans**

- Describe each *aligned* particle as a reduced set of a few features.
- Group particles with similar features.

**Multi Reference Alignment**

- Align particles to several references and compare fitting quality.
- Alignment and classification occur simultaneously.
PCA Classification: Distance matrix:

Missing wedge-aware cross-correlation coefficients for all possible particle couples

Element $m(i,j)$ similarity of particle $i$ and particle $j$

Constrained cross correlation
both particles need to be filtered to their common Fourier components.

Two noise-free but differently rotated copies of the same particle should have maximum cross correlation coefficient...

... but they are differently affected by the missing wedge....

... directly comparing them after Alignment will yield a poor score....

both particles need to be filtered to their common Fourier components.
Hierarchical clustering

... is the most trivial approach, and ....never works with real data:
  it only sees a global distance between particles
Diagonalize similarity matrix:

Why?

Eigenvolumes:
Orthogonal to each other.

Eigenvalues:
Percentage of explained variance

Not always easy to give an intuitive physical interpretation
Diagonalization

- A systematic way to create eigenvolumes as linear combinations of particles:
- Very well known, stable, efficient numerical procedure.

$L$ eigenvectors (eigenparticles) $<< \quad N$ particles
Each particle can be expressed as a vector of a few scalars

\[ \approx \lambda_{i,1} \cdot + \lambda_{i,2} \cdot + \lambda_{i,3} \cdot + \lambda_{i,4} \cdot \]

\[ \approx \lambda_{2,1} \cdot + \lambda_{2,2} \cdot + \lambda_{2,3} \cdot + \lambda_{2,4} \cdot \]

\[ \approx \lambda_{3,1} \cdot + \lambda_{3,2} \cdot + \lambda_{3,3} \cdot + \lambda_{3,4} \cdot \]

\[ \approx \lambda_{N-2,1} \cdot + \lambda_{N-2,2} \cdot + \lambda_{N-2,3} \cdot + \lambda_{N-2,4} \cdot \]

\[ \approx \lambda_{1N-1,1} \cdot + \lambda_{1N-1,2} \cdot + \lambda_{1N-1,3} \cdot + \lambda_{1N-1,4} \cdot \]

\[ \approx \lambda_{N,1} \cdot + \lambda_{N,2} \cdot + \lambda_{N,3} \cdot + \lambda_{N,4} \cdot \]

\[ \lambda_{i,j} : \text{“weight” of eigenvolume j in particle i} \]
Each point represents a particle in a L-dimensional space

K-means: clustering in L-dimensional spaces provided by L Principal Components
Inspect/statistically test the histograms of each eigencomponent:

- Discard gaussian distributed components.
- Identify very salient features.

Use a priori information: symmetry order.

Repeat experiments with different masks.

Use all the mathematical machinery for clustering available from the DS community.
Pitfalls:

Avoid classifying according to orientations!

$\theta$ angle (orientation of particle with respect to the beam)

$\lambda_3$,

$\text{Eigenvolume #3}$

$\theta$ angle (orientation of particle with respect to the beam)
Multi Reference Alignment: simultaneous alignment and classification.

Align all particles against several 3d templates. Particles will contribute only to the best fitting template.

Template #1 aligns all particles.

low cross correlation  

high cross correlation
Multireference Alignment (MRA)
Simultaneous alignment and classification.

~8000 vertices

Each red box represents the projection of a single vertex particle along y

Just adding the particles:

y-view
z-view
Refining produces more structure.....

z-view (single slice)

y-view (all slices of average)

Red: extent of the mask used for alignment
.... but refining with a classification mask placed outside the capsid reveals that some particles (vertices) have a spike...

5% of the data set

with Nicola Abrescia, Biogune (Spain)
Data management
Idealized pipeline for subtomogram averaging

1. tilt series
   - alignment
2. aligned tilt series
   - reconstruction
3. tomogram
   - density map of imaged sample
   - particle identification
4. subtomograms
   - density maps of individual particles
   - averaging
5. density map
• Several volumes (tens... or even hundreds)
• Several modeling steps might be involved, both (semi) automatic or user driven.
Goal setting at the first stage of Dynamo: just Subtomogram Averaging

- Design and run lots of different experiments
- Create once a fixed set of data particles

+ Alignment parameters
Explicit management
- command line orders
- native scripts
- dedicated GUIs

Automated import
- tomogram browsers
- geometric design

Catalogue

Project (refinement)
Creating a Catalogue

simple text file listing tomograms

tomogram1.mrc
tomogram2.mrc
tomogram3.mrc

>> dcm -create catalogue -vll files.vll
exploration of sets of tomograms
Tomograms can be large!! → memory control

- preview tool for disk contents
- selection or regions of interest
- Cataloguing of r.o.i.
Ok, I just opened one tomogram... how do I get my particles?

Different structures with different inherent geometry:
- microtubules
- vesicles
- bent filaments
- membranes
- isolated particles...

Different models
- OO framework: class model, subclasses for filaments, membranes, etc

The same tomogram visualization GUIs adapt their interaction schemes to the different models
Modeling example: 
proteins in cellular membranes 
(geometric operations for triangulations)

INPUT: Membrane points 
\textit{clicked, imported, detected}.....

Associated model tools: 
- spline fitting
- triangulation (Crust)
- subdivision (Loop)
- geodesic partition
- formatting for \textit{Dynamo StA}

Intended output 
Model a membrane geometry to infer: 
- Regular distribution of particle locations. 
- Initial orientations (normal to membrane).
• Particles in boxes.
• First template.
• Initial orientations.

You’re in business for subtomogram averaging!

... and all elements are already formatted for Dynamo StA
Modeling of tubular objects
Densely packed filaments

...... may have a very convoluted geometry

.... requiring visualization and picking along natural geometry.

Automated refinement
Additional modeling tricks: *On screen selection of isolated particles*
High Performance Computing
The problem allows a simple parallelization strategy:

1 – Coarse grain parallelization

algorithm is “embarrassingly parallelizable”

D particles

T independent processors

T subsets of data
(D/T) particles each

T threads / tasks

2 – Fine grain parallelization

adapt alignment procedure for GPU execution

1/O, basic settings
divide particles
into subsets

alignment

subset 1

subset 2

subset T

thresholding

averaging

wedge weighting

sequential I/O and alignment

sequential I/O and alignment

sequential I/O and alignment

iteration (n)

setup

compute

assemble

parallelization strategy:

T subsets of data
(D/T) particles each

T threads / tasks

D particles

T independent processors

T cores in a cluster (DM)
T cores in a server (SM/DM)
T GPUs in a server
T GPUs in a cluster

algorithm is “embarrassingly parallelizable”

adapt alignment procedure for GPU execution
What is a GPU?

The Graphic Processing Unit is a co-processor specialized on mathematical operations required by graphical rendering (“the graphics pipeline”)

What is GPU computing?

*Using dedicated GPUs for non-graphical tasks. Needs creation of GPU-specific code.*

*Architecture is optimized for compute intensive tasks*

*Slow memory bandwidth: not appropriate for data intensive tasks*
GPU computing for subtomogram averaging

The core task in subtomogram averaging refinement is extremely GPU friendly!

1. Fetch a particle, template and mask from hard disk into memory.  
   *Takes a lot of time... but you do it once per particle and iteration.*

2. Transfer particle, template and mask from memory to GPU.  
   *Takes a lot of time... but you do it once.*  
   *Memory in the GPU is pricey... but you don’t need a lot (~Mb)*

3. Lots of rotations and Fourier transforms.  
   *And GPUs love this!*  
   *Speed-up factors [CPU-GPU] between 8x and 40x*
GPU computing in 2015

Great times!!

API revolution

Creation of GPU code has become much easier in the last ten years:
CUDA 7.0 : unified memory addressing system for CPU and GPU.
Matlab/Python libraries for direct talk to GPU.

Cost revolution

GPUs themselves are way cheaper than CPU clusters.

NVIDIA introduced the Titan Black consumer card in 2013
High end K40c ≈ 4000$  <->  Titan black ≈ 500-800$
Comparable performance (at the cost of some “butts” ....)
High performance computing: classification

In PCA, the main bottleneck is the computation of the similarity matrix.
After a disk access, a data particle undergoes just a Fourier transform and a rotation.

1. GPU computing will NOT provide any speed up!
2. Selection of number of particles processed in a single block is important.

- Too less particles:
  - Same particles are being read from disk continuously.

- Too many particles:
  - Excessive memory consumption can crowd and crash the core.
Flagellar motors of *Borrelia* spirochetes

used by bacteria to generate the rotation of flagellar filaments

Raddi et al, Bacteriology, 2012
Alignment:

round 1:  8 cycles on 1x binned data  (20min  min on 4 GPU Tesla /  65 hour in one CPU)
round 2:  8 cycles  on unbinned data  (3 hours of 4 GPU Tesla /  21 days in one CPU)
y-slices on the average of 50% best particles (according to CC coefficient)
Possible hint for heterogeneity?

Different intensities
matrix computation + PCA + Kmeans + class refinement:

8 minutes (6 CPU) 12 minutes (1 CPU) 1 hour (3 GPU)

Two main classes are clearly identifiable
Subboxing: separate alignment of asymmetrical units
Want to learn more about subtomogram averaging?

Come to the Dynamo Workshop 2015 in Basel this August!

www.dynamo-em.org

Participants of the annual Dynamo workshop 2012, Basel, Switzerland
THANKS!
Back up slides
- Nicola Abrescia, Biogune (Bilbao)
- Giulia Zanetti, Birkbeck (London)
- Alex Noble, Florida State University
- Juha Huiskonnen, Oxford University
- Saskia Bakker, Glasgow University
- Ashraf al-Amoudi, DZNE Bonn
- Sai Li, Oxford University
- Florian Schur, EMBL Heidelberg
- Max Maletta, NKI Cancer Institute (Amsterdam)
- John Driscoll, Birkbeck, London
- Andrzej Miezczak, MPI Dresden
- Ludo Renault, Cancer Institute, London
- Morgan Beeby, Imperial College London
- Gaia Pigino, MPI Dresden
- Bara Malkova, PSI Zurich
- Chi-Yu Fu, Scripps Institute (San Diego)
- Karen Davies, Frankfurt University
- James Carpino, New York University
- Peter Engelhardt, Helsinki University
- Daven Vashestan, Oxford University
- James Streetley, MRC London

(and ~250 registered users)
Online case study: Type III secretion system in Yersinia bacterium


Tomograms

Models

Average

Subtomograms

Further studies: i.e. classification

Different elongation states
Goal I: Resolution improvement

The original sin of subtomogram averaging
Goal: Resolution improvement

The original sin of subtomogram averaging

What if the subtomogram itselfs are not correctly aligned?

Let’s introduce an error that simulates the typical error introduced onto the tomogram during tilt series alignment.
No compensation of tilt series alignment error through refinement process:

“perfect” particles

“shift affected” particles

No matter how you do your StA, you are stuck with the tomographic alignment error
Alignment

enforce coherent signal contributions

Thresholding

identify coherent signal contributions

Averaging

enhance signal

GIVEN
reference \( q^{(n)} \) data set \( \{ d_i \}_i \)
mask \( M \), similarity function \( s_M (\cdot, \cdot) \)
a set of orientations \( T_i^{(n)} \) for each \( i \)

DO for each \( d_i \):
DO for each orientation \( T_j \in T_i^{(n)} \)
compute \( s_M (T_j^{-1}q^{(n)}, d_i) \)
\( T_i := \text{arg max}_{T_j \in T_i^{(n)}} s_M (T_j^{-1}q^{(n)}, d_i) \)

THRESHOLDING

DO for each \( d_i \):
DO for each orientation \( T_j \in T_i^{(n)} \)
compute \( s_M (T_j^{-1}q^{(n)}, d_i) \)
\( T_i := \text{arg max}_{T_j \in T_i^{(n)}} s_M (T_j^{-1}q^{(n)}, d_i) \)

GIVEN
reference \( q^{(n)} \) data set \( \{ d_i \}_i \)
masks \( M, M' \), similarity function \( s_M (\cdot, \cdot) \)
a dynamically defined set \( T_i^{(n)} \) for each \( i \)

DO for each \( d_i \):

Thresholding

GIVEN
reference \( q^{(n)} \) data set \( \{ d_i \}_i \)
masks \( M, M' \), similarity function \( s_M (\cdot, \cdot) \)
a dynamically defined set \( T_i^{(n)} \) for each \( i \)

Alignment

Thresholding

Averaging

\[
 q^{(n+1)} = \frac{1}{\# I} \sum_{i \in I} T_i d_i 
\]

\[
 q^{(n+1)} = \frac{1}{\# I} \sum_{i \in I} T_i d_i 
\]

\[
 q^{(n+1)} = \frac{1}{\# I} \sum_{i \in I} T_i d_i 
\]

\[
 q^{(n+1)} = \frac{1}{\# I} \sum_{i \in I} T_i d_i 
\]
1 – Coarse grain parallelization
algorithm is "embarrassingly parallelizable"

D particles
T independent processors
T cores in a cluster (DM)
T cores in a server (SM/DM)
T GPUs in a server
T GPUs in a cluster

T subsets of data
(D/T) particles each
T threads / tasks

2 – Fine grain parallelization
adapt alignment procedure for GPU execution

Part III: High Performance Computing
Dynamo (Part III): High Performance Computing

1 – Coarse grain parallelization
algorithm is “embarrassingly parallelizable”

D particles
T independent processors

T subsets of data (D/T) particles each
T threads / tasks

T cores in a cluster (DM)
T cores in a server (SM/DM)
T GPUs in a server
T GPUs in a cluster

2 – Fine grain parallelization
adapt alignment procedure for GPU execution

I/O, basic settings divide particles into subsets

thresholding averaging wedge weighting

setup
compute
assemble

MPI/thread manager

thread 1
thread 2
thread T

subset 1
subset 2
subset T

sequential I/O and alignment
sequential I/O and alignment
sequential I/O and alignment
## Part III: High Performance Computing

<table>
<thead>
<tr>
<th>Platform</th>
<th>Computing system</th>
<th>Performance</th>
<th>CINA resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicore desktop/server</td>
<td>OpenMP or Posix, MPI (planned)</td>
<td>not OK 8 cores give 2x gain</td>
<td>3 servers 8 cores each</td>
</tr>
<tr>
<td>CPU cluster</td>
<td>MPI + Matlab MCR</td>
<td>OK almost linear scaling</td>
<td>600 cores (Biozentrum) 22000 cores (CSCS)</td>
</tr>
<tr>
<td>GPU</td>
<td>CUDA+Matlab or CUDA+Matlab MCR</td>
<td>OK 10x to 20x</td>
<td>3 Tesla + 1 Geforce</td>
</tr>
</tbody>
</table>

All (reasonable) flavours of HPC are supported transparently for the user.
... producing a very fast evolving panorama....

<table>
<thead>
<tr>
<th>GPU project</th>
<th>speedup</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonlinear Diffusion</td>
<td>20x</td>
<td>none!</td>
</tr>
<tr>
<td>(Strozdka et al, IEEE-CV, 2001)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterative reconstruction</td>
<td>70x</td>
<td>SH</td>
</tr>
<tr>
<td>(SIRT, SART, ART...)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Castaño-Díez et al, JSB, 2006)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>General EM tools</td>
<td>2x-90x</td>
<td>CUDA 1.0</td>
</tr>
<tr>
<td>(filtering, back projection...)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Castaño-Díez et al, JSB, 2008)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allignator</td>
<td>8x-12x</td>
<td>Matlab + Jacket</td>
</tr>
<tr>
<td>Markerless Alignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Castaño-Díez et al, JSB, 2010)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamo</td>
<td>10x-60x</td>
<td>CUDA 5.0</td>
</tr>
<tr>
<td>Subtomogram Averaging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Castaño-Díez et al, JSB, 2012)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Given

Reference $q^{(n)}$ data set $\{d_i\}_i$

Masks $M, M^c$, similarity function $s_M(\cdot, \cdot)$

A dynamically defined set $T_i^{(n)}$ for each $i$

Do for each $d_i$:

Do for each orientation $T_j \in T_i^{(n)}$

Compute $s_M(T_j^{-1}q^{(n)}, d_i)$

$T_i := \arg \max_{T_j \in T_i^{(n)}} s_M(T_j^{-1}q^{(n)}, d_i)$

$\tilde{q}^{(n)} = \frac{1}{\#T_i} \sum_{i \in \tilde{I}} T_i d_i$

$\tilde{I} = \{i : s_{M^c}(\tilde{q}^{(n)}, T_i d_i) > \tau\}$

$q^{(n+1)} = \frac{1}{\#\tilde{I}} \sum_{i \in \tilde{I}} T_i d_i$

Iteration

Alignment

Do for $i$:

Align particle $i$

Averaging

Iteration