Segmentation for Diffraction Contrast Tomography

Seminar @ Centre de Morphologie Mathématique MINES ParisTech

João P C Bertoldo

Materials Center @ MINES Paristech - PSL University
ID11 @ The European Synchrotron Radiation Facility (ESRF)

31 May 2021

Diffraction Contrast Tomography (DCT)

Recall: "usual" tomography

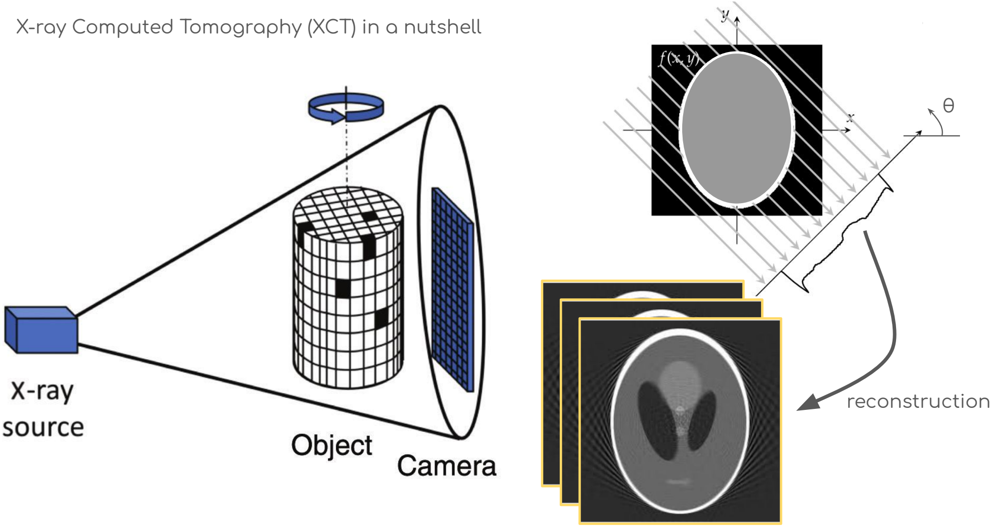

Figure 1: XCT recall

Diffraction Contrast Tomography (DCT)

A short introduction

- Multiple 2D images acquired from different angles
- Acquire transmitted and diffracted beam

  (notice: the acquisition area is larger than the beam)
- Reconstruct each grain individually from the diffraction spots

← click

Figure 2: DCT setup (simulation). Credits: Wolfgang Ludwig.

Diffraction Contrast Tomography (DCT)

What is it useful for?

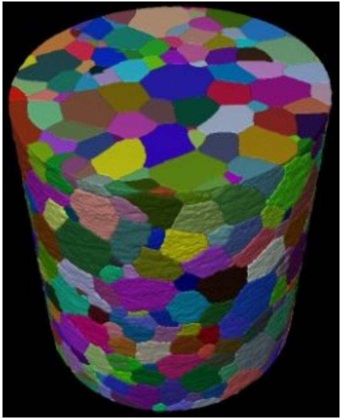

Figure 3: Reconstructed volume with individual grains segmented (different colors). Credits: Wolfgang Ludwig.

- Individual grains naturally segmented
- Grain's plane direction
- Slip bands visualization (?)

Pipeline: from experiment to 3D digital twin

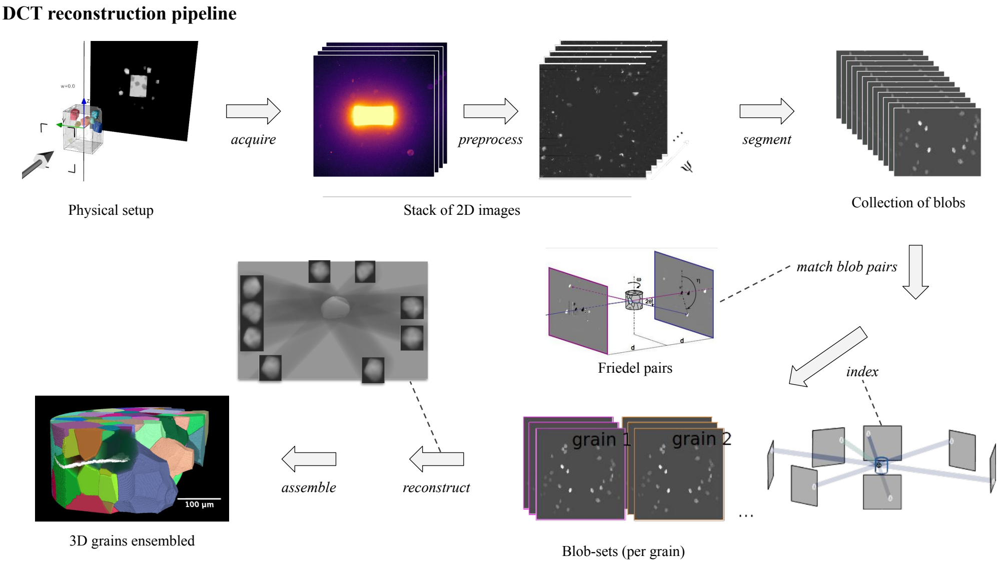

Figure 4: main steps in DCT's reconstruction pipeline

Pipeline: from experiment to 3D digital twin

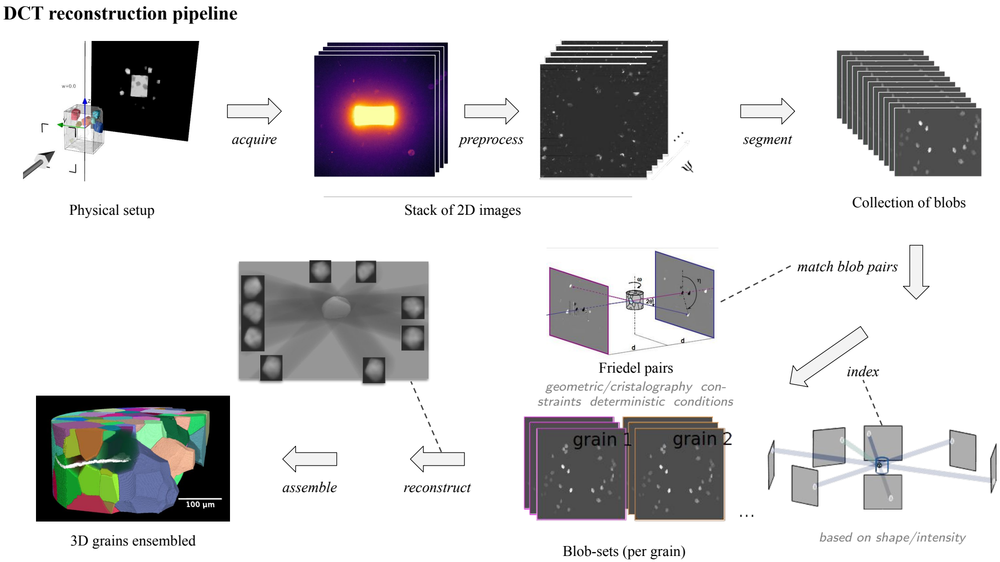

Figure 4: main steps in DCT's reconstruction pipeline

Pipeline: from experiment to 3D digital twin

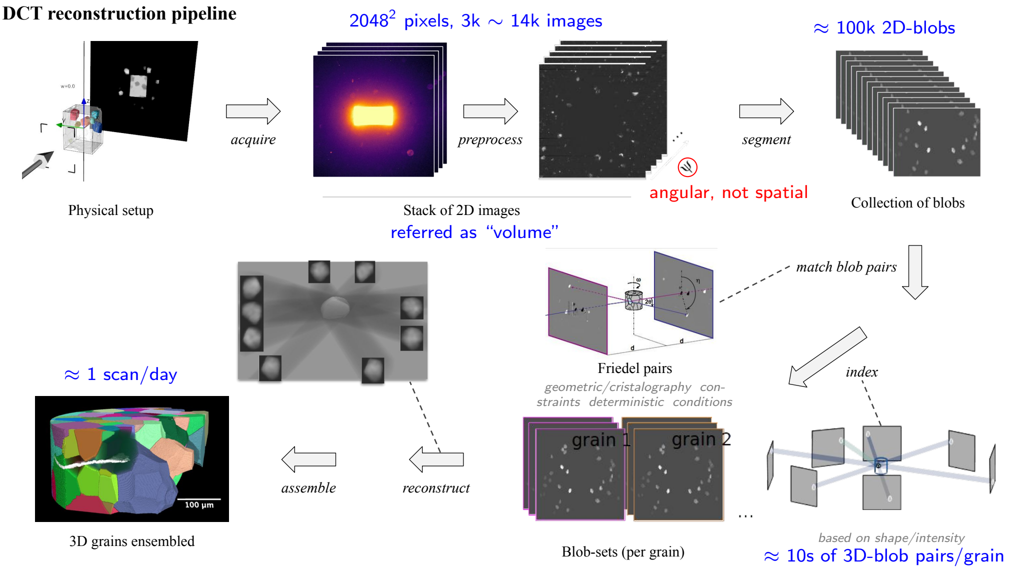

Figure 4: main steps in DCT's reconstruction pipeline

Pipeline: from experiment to 3D digital twin

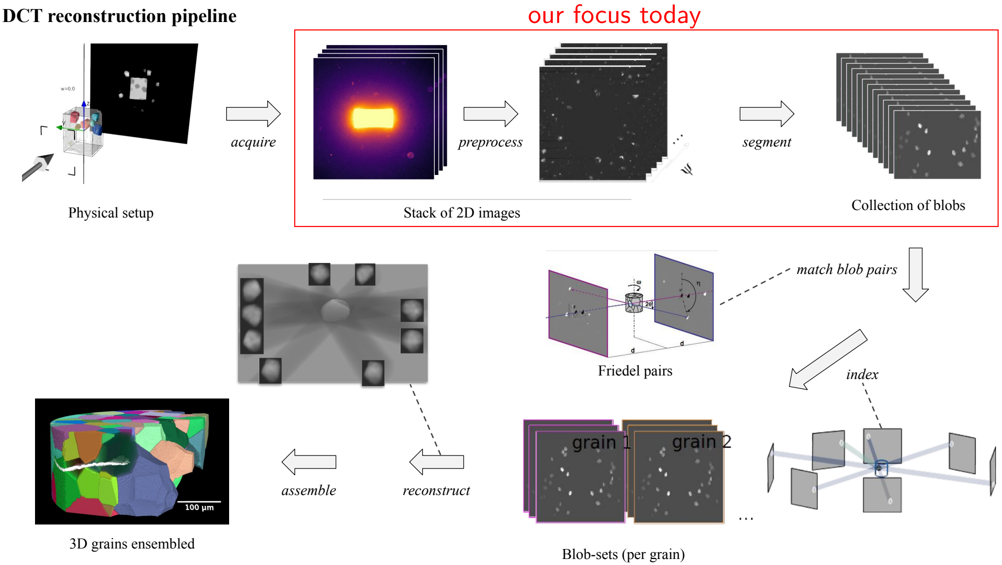

Figure 4: main steps in DCT's reconstruction pipeline

Raw image and preprocessing steps

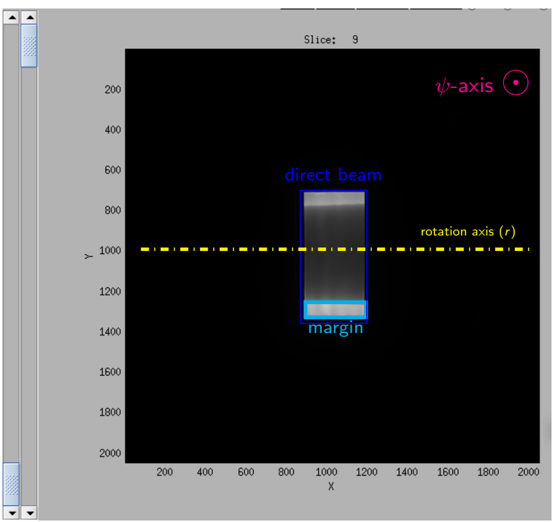

Figure 5: raw acquisition (increased contrast)

Image characteristics

- (sometimes) 1 image = 4-image grid
- source intensity oscillates

  i.e. background variation (over ψ)

Pre-processing steps

- subtract offset ("dark image")
- (manually) select direct beam and margin
- normalize by the margin's average

Raw image and preprocessing steps

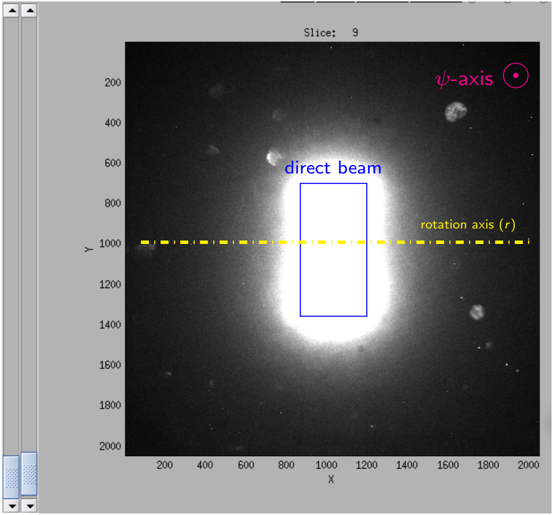

Figure 5: raw acquisition (increased contrast)

Image characteristics

- (sometimes) 1 image = 4-image grid
- source intensity oscillates
  i.e. background variation (over ψ)
- beam/blob brightness 2 scales apart

Pre-processing steps

- subtract offset ("dark image")
- (manually) select direct beam and margin
- normalize by the margin's average

Raw image and preprocessing steps

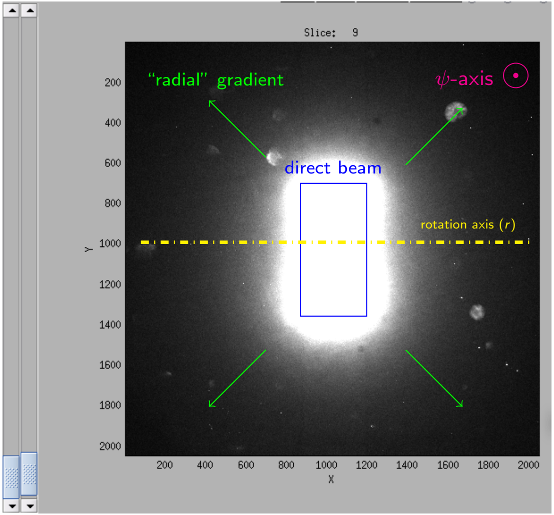

Figure 5: raw acquisition (increased contrast)

Image characteristics

- (sometimes) 1 image = 4-image grid
- source intensity oscillates
  i.e. background variation (over ψ)
- beam/blob brightness 2 scales apart
- "radial" blur superposed on blobs

Pre-processing steps

- subtract offset ("dark image")
- (manually) select direct beam and margin
- normalize by the margin's average
- subtract a per-pixel ψ-wise moving median

Raw image and preprocessing steps

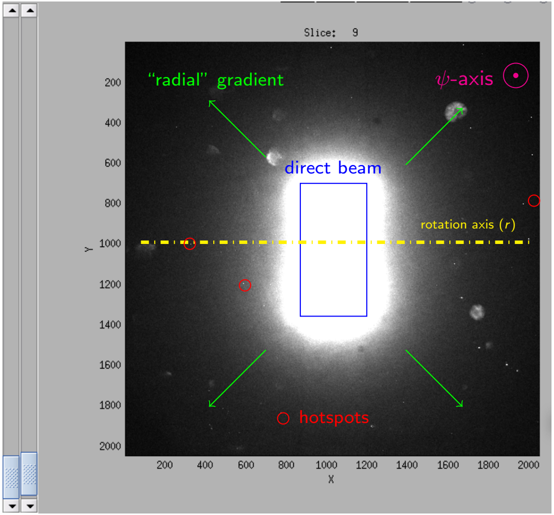

Figure 5: raw acquisition (increased contrast)

Image characteristics

- (sometimes) 1 image = 4-image grid
- source intensity oscillates
  i.e. background variation (over ψ)
- beam/blob brightness 2 scales apart
- "radial" blur superposed on blobs
- hotspots all over
- blobs closer to r are ψ-longer

Pre-processing steps

- subtract offset ("dark image")
- (manually) select direct beam and margin
- normalize by the margin's average
- subtract a per-pixel ψ-wise moving median
- 2D median filter (per frame)

ψ-wise moving median

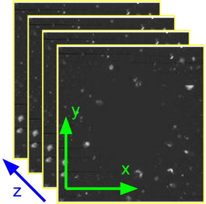

Figure 6: stack of 2D images
typo: ψ, not z

3D image

- W: width, along x, indexed by $i \in \{1 \ldots W\} = I$
- H: height, along y, indexed by $j \in \{1 \ldots H\} = J$
- D: depth[a], along ψ, indexed by $k \in \{1 \ldots D\} = K$
- $\mathcal{I}$: image, a 3D grid $\in [0, V]^{(W,H,D)}$, where V is a constant

[a] i.e. nb. of 2D frames, i.e. nb of rotation

ψ-WISE MOVING MEDIAN

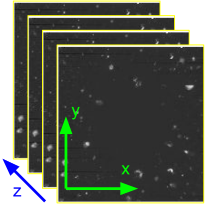

Figure 6: stack of 2D images
typo: ψ, not z

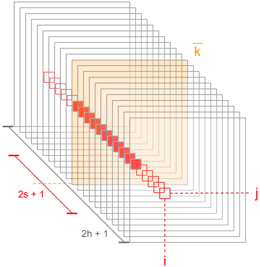

Figure 7: pixel-wise moving median

3D IMAGE

- W: width, along x, indexed by $i \in \{1 \ldots W\} = I$
- H: height, along y, indexed by $j \in \{1 \ldots H\} = J$
- D: depth[a], along ψ, indexed by $k \in \{1 \ldots D\} = K$
- $\mathcal{I}$: image, a 3D grid $\in [0, V]^{(W,H,D)}$, where V is a constant

[a] i.e. nb. of 2D frames, i.e. nb of rotation

BACKGROUND

$$\mathcal{B}_{i,j,\overline{k}} = \underset{k' \in range(\overline{k},w)}{\text{median}} \mathcal{I}_{i,j,k'}$$

where w is a parameter (typically 500) and
$range(\overline{k}, w) = \{k\prime \in \mathbb{Z} \mid max(0, \overline{k} - w) \leq k\prime \leq min(D, \overline{k} + w)\}$

ψ-wise moving median

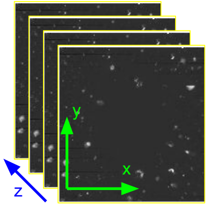

Figure 6: stack of 2D images

typo: ψ, not z

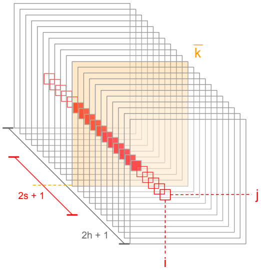

Figure 7: pixel-wise moving median

3D image

- W: width, along x, indexed by $i \in \{1 \ldots W\} = I$
- H: height, along y, indexed by $j \in \{1 \ldots H\} = J$
- D: depth[a], along ψ, indexed by $k \in \{1 \ldots D\} = K$
- $\mathcal{I}$: image, a 3D grid $\in [0, V]^{(W,H,D)}$, where V is a constant

[a] i.e. nb. of 2D frames, i.e. nb of rotation

Background

$$\mathcal{B}_{i,j,\overline{k}} = \underset{k' \in range(\overline{k},w)}{\text{median}} \mathcal{I}_{i,j,k'}$$

where w is a parameter (typically 500) and
$range(\overline{k}, w) = \{k' \in \mathbb{Z} \mid max(0, \overline{k} - w) \leq k' \leq min(D, \overline{k} + w)\}$

Image - background

For all $(i,j) \in I \times J$ and $\overline{k} \in \{0, \ldots, \lceil \frac{K}{2s+1} \rceil\}$:

$$\mathcal{I}^*_{i,j,k'} = \mathcal{I}_{i,j,k'} - \mathcal{B}^h_{i,j,\overline{k}} \quad \text{for all } k' \in range(\overline{k}, s)$$

where s is a parameter (typically 50).

ψ-wise moving median

Remarks

- Notice: the method is adapted to the radial gradient because the medians are pixel-wise; xy-operations/filtering could be worse because of the overlapped glow.
- Parameters: $h = 250$ and $s = 25$
- $W = H \approx 2.000 \rightarrow 4.000.000$ pixels/background
- $D \approx 10.000 \rightarrow 10.000/2s = 10.000/50 = 200$ backgrounds
- $4.000.000 \times 200 \approx 10^9$ medians of 500 values each...

Solutions attempted in Python (on CPU)

- Ref: (mono-core) `numpy.median` vectorized on XY
- Ref + numba on 16 cores: speedup factor ≈ 10
- Ref + multiprocessing on 16 cores : speedup factor ≈ 40
- other ad-hoc ideas much slower (complexity + python execution)

Preprocessed image

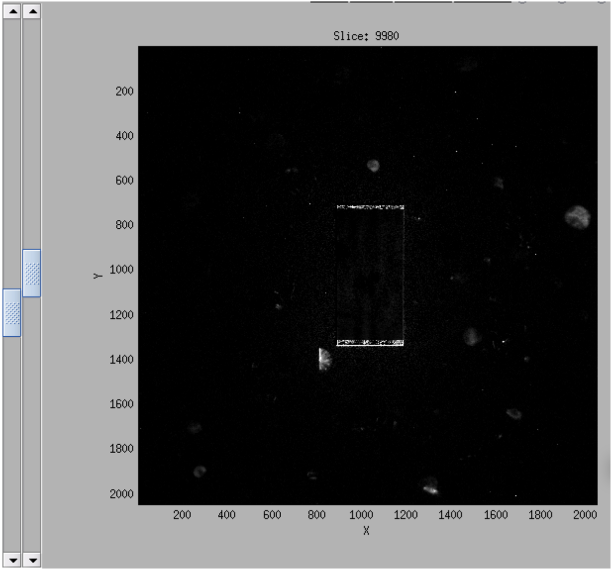

Figure 8: preprocessed DCT image

Processing time

Ex.: 7200-frame image:

- Previous (MATLAB)
  - 8 machines
  - ≈ 30 minutes
- Now (Python):
  - 1 machine (16 cores)
  - ≈ 15 minutes
  - 1/3: shared-memory allocation + data copy
  - 1/3: computations
  - 1/3: load/save data

Segmentation

Double threshold in 1D

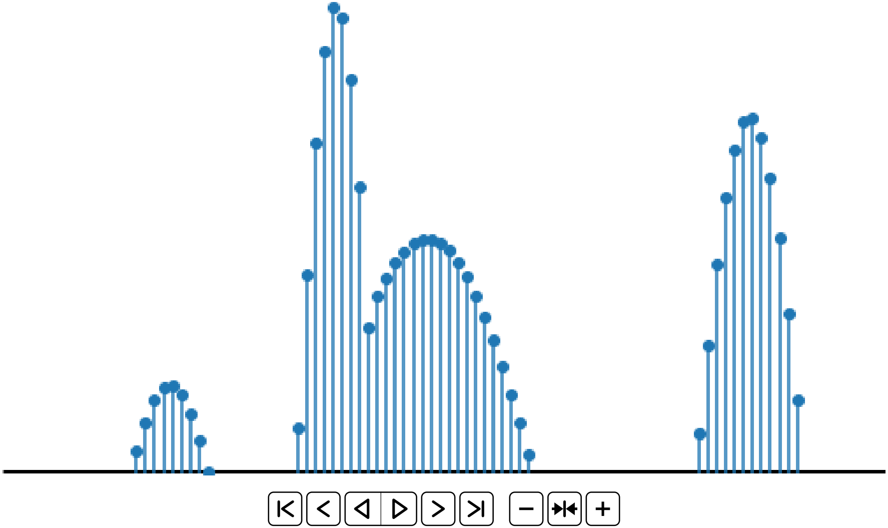

Figure 9: double thsrehold illustration

Segmentation

Double threshold in *3D*

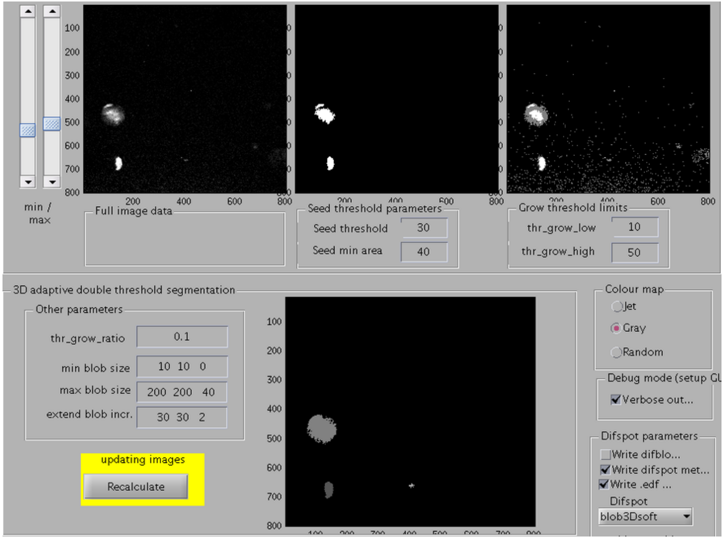

Figure 10: Double threshold interface

- pick a *seed* (minimum) threshold
- find all connected regions (blobs)
- for each blob:

Segmentation

Double threshold in *3D*

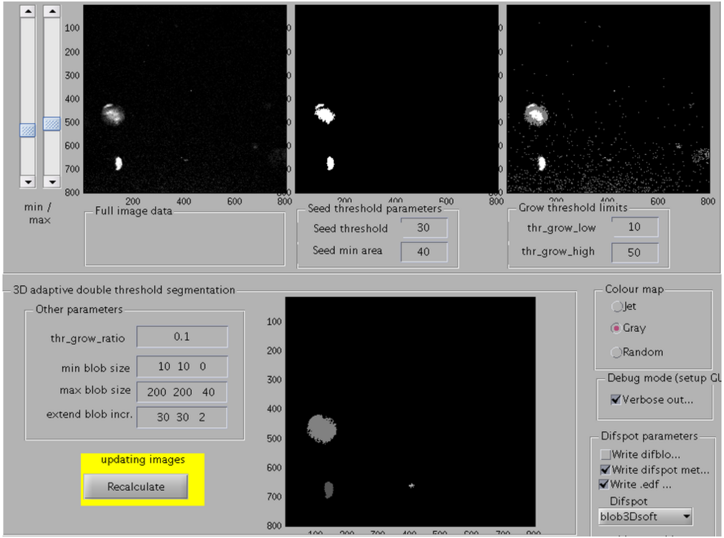

Figure 10: Double threshold interface

- pick a *seed* (minimum) threshold
- find all connected regions (blobs)
- for each blob:
- pick a tolerance $\tau \in [0, 1]$ and clipping limits c_{min} and c_{max}
- for each blob:
  - find local maximum M
  - define a range $R = [max(c_{min}, \tau M), min(c_{max}, M)]$
  - iteratively grow the region with neighbor voxels $\in R$

Segmentation

Double threshold in *3D*

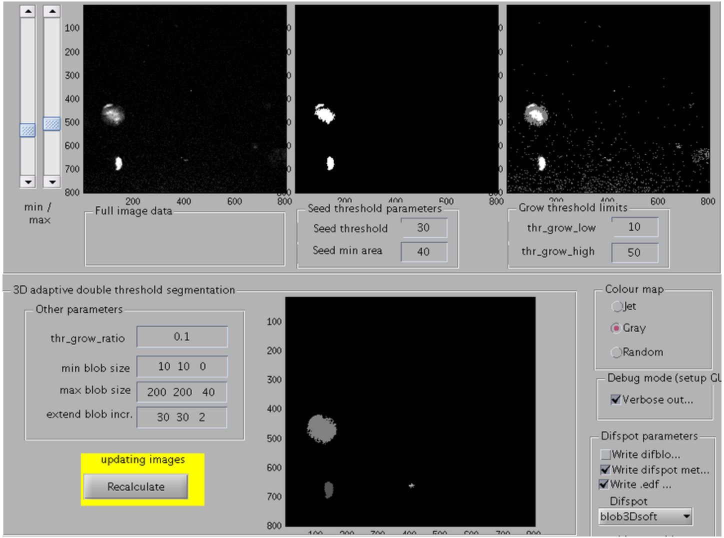

Figure 10: Double threshold interface

- pick a *seed* (minimum) threshold
- find all connected regions (blobs)
- for each blob:
- pick a tolerance $\tau \in [0, 1]$ and clipping limits c_{min} and c_{max}
- for each blob:
  - find local maximum M
  - define a range $R = [max(c_{min}, \tau M), min(c_{max}, M)]$
  - iteratively grow the region with neighbor voxels $\in R$
- pick clipping bounding box sizes $bb_{min}, bb_{max} \in \mathbb{R}^3$
- discard blobs too big or too small

Segmentation

Double threshold in *3D*

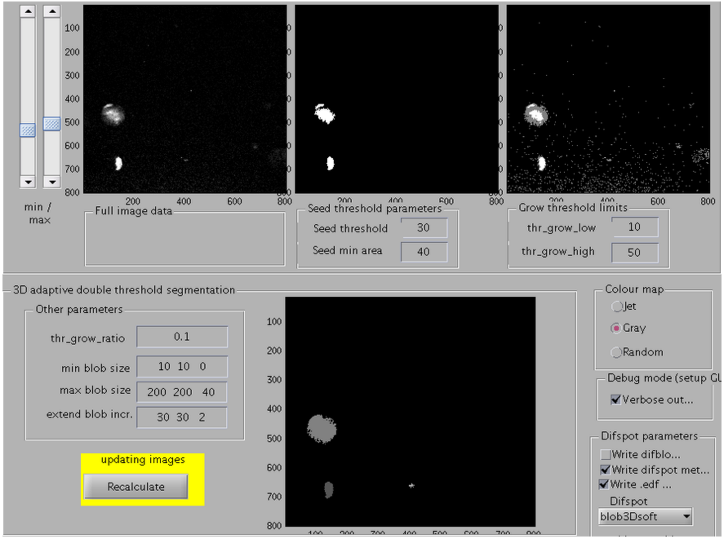

Figure 10: Double threshold interface

- pros
  - ψ-length discards alongated diffractions close to the rotation axis
  - x and y-length clean noisy blobs
  - relatively simple

Segmentation

Double threshold in *3D*

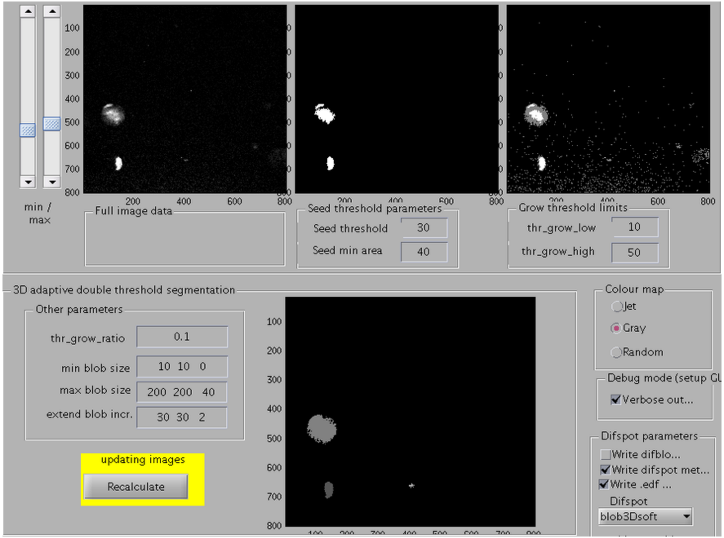

Figure 10: Double threshold interface

- pros
  - ψ-length discards alongated diffractions close to the rotation axis
  - x and y-length clean noisy blobs
  - relatively simple
- cons
  - slow
    - iterative region growing
    - $\approx 10^4$ blobs
    - a few hours on ≈ 10 jobs
  - lots of (manual) parameters
  - which are picked based on a sub-volume

Segmentation

Double threshold in *3D*

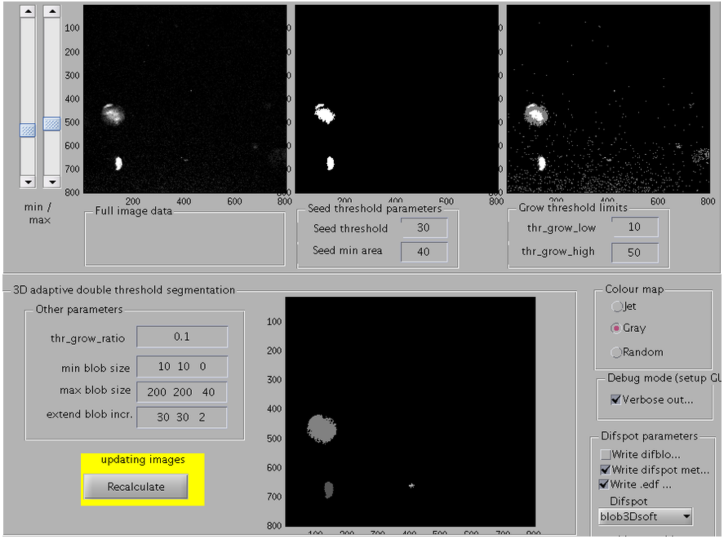

Figure 10: Double threshold interface

- pros
  - ψ-length discards alongated diffractions close to the rotation axis
  - x and y-length clean noisy blobs
  - relatively simple
- cons
  - slow
    - iterative region growing
    - $\approx 10^4$ blobs
    - a few hours on ≈ 10 jobs
  - lots of (manual) parameters
  - which are picked based on a sub-volume
- next
  - re-implement in Python
  - let's see what a U-net can do

Thank you!

Joao P C Bertoldo

Materials Center @ MINES Paristech - PSL University
ID11 @ The European Synchrotron Radiation Facility (ESRF)

31 May 2021