Sep 05, 2024

Tomogram Reconstruction and Gold Fiducial Removal with Warp, Etomo, and Fidder

DOI
dx.doi.org/10.17504/protocols.io.6qpvr8qbblmk/v1

Connor Garrels[1], Benjamin Barad[1], Steve Reichow[1]

[1]Oregon Health & Science University

Connor Garrels
Oregon Health & Science University

DOI: dx.doi.org/10.17504/protocols.io.6qpvr8qbblmk/v1

Protocol Citation: Connor Garrels, Benjamin Barad, Steve Reichow 2024. Tomogram Reconstruction and Gold Fiducial Removal with Warp, Etomo, and Fidder. **protocols.io** https://dx.doi.org/10.17504/protocols.io.6qpvr8qbblmk/v1

Protocol status: Working
We use this protocol and it's working

Created: August 19, 2024

Last Modified: September 05, 2024

Protocol Integer ID: 105958

Keywords: cryoET, tomography, image processing, python

Abstract

While gold nanoparticle fiducials greatly improve tilt-series alignment in cryo-electron tomography data, they can cause artifacts during later image processing steps. We present a workflow for high-quality fiducial removal using fidder, an open-source python package. This protocol demonstrates how to flexibly incorporate fidder into tomogram reconstruction using Warp and Etomo.

Overview

1 Gold nanoparticle fiducials in cryoET can greatly improve tilt-series alignment and tomogram quality. They can also introduce detrimental artifacts in downstream processing steps such as denoising or membrane segmentation. While there are methods to remove gold beads from micrographs, these methods are not always high quality or easily integrated into existing workflows.

Here, we describe a workflow using fitter as part of tomogram reconstruction with Warp and Etomo, incorporating a step for gold fiducial removal with Fidder. The resulting tomograms are high quality because of the ability to use gold during alignment, but with no visible gold artifacts remaining.

Install Software

2 Make sure you have these packages installed.

- **IMOD**
- **Warp** (technically WarpTools on linux)
- **Fidder**

Warp Preprocessing

3 We will be using Warp for the bulk of our processing. This section follows **this warp tutorial**.

Values shown are data-specific (i.e. exposure, pixel size, paths), so please be careful when copying these commands. These are just example values!

3.1 **Create Settings Files**

frameseries

```
WarpTools create_settings --folder_data /where/your/movies/are --
folder_processing frameseries --output frameseries.settings --
extension *.tif --angpix 0.8125 --bin 1 --gain_path
/where/your/gain/is --exposure 3
```

tiltseries

```
WarpTools create_settings --output tiltseries.settings --
folder_processing tiltseries --folder_data tomostar --extension
*.tomostar --angpix 0.8125 --bin 1 --gain_path /where/your/gain/is
--exposure 3 --tomo_dimensions 9600x12000x2500
```

If your data was collected at super-resolution like ours was, --bin 1 is very important!

3.2 **Motion Correction and CTF Estimation**

Motion correct and CTF estimate your data and output whole frame averages as well as even and odd frame-split micrographs for later denoising.

```
WarpTools fs_motion_and_ctf --settings frameseries.settings --
m_grid 8x6x10 --c_grid 2x2x1 --c_range_max 7 --c_defocus_max 8 --
c_use_sum --out_averages --out_average_halves
```

3.3 **Import tiltseries metadata**

```
WarpTools ts_import --mdocs /where/your/mdocs/are --frameseries
frameseries --tilt_exposure 3 --min_intensity 0.3 --dont_invert --
output tomostar
```

You may need to edit your mdocs to reflect where your motion corrected micrographs are. If so, try this script.

```
#!/usr/bin/env python3

import os

mdoc_folder = '/where/your/mdocs/are'
new_subframe_path = 'where/your/micrographs/are'

def update_subframe_path(mdoc_path, new_path):
    with open(mdoc_path, 'r') as f:
        lines = f.readlines()

    for i, line in enumerate(lines):
        if line.startswith('SubFramePath'):
            existing_path = line.split('=')
[-1].strip().replace('\\', '/')
            _, existing_filename = os.path.split(existing_path)

            new_subframe_path = f'SubFramePath =
{os.path.join(new_path, existing_filename)}\n'
            lines[i] = new_subframe_path

    with open(mdoc_path, 'w') as f:
        f.writelines(lines)

def batch_update_paths(folder_path, new_path):
    for filename in os.listdir(folder_path):
        if filename.endswith('.mdoc'):
            mdoc_path = os.path.join(folder_path, filename)
            update_subframe_path(mdoc_path, new_path)
            print(f'SubFramePath updated for {filename}')

batch_update_paths(mdoc_folder, new_subframe_path)
```

3.4 **Make Stacks**

```
WarpTools ts_stack --settings tiltseries.settings --angpix 6.5
```

Notice we chose to make our stacks at 6.5 Å (bin 4 from physical pixel size). Whatever size you choose, remember it, as it is the resolution you will use for alignment and reconstruction later.

Etomo tilt series alignment

4 We will use batch processing in etomo for aligning our newly created stacks. Components of this section follows **this warp tutorial.**

4.1 **Make etomo directory**
Create the directory 'etomo' in your /tiltseries directory.

4.2 **etomo**

Open etomo in your /tilseries/etomo directory and choose the 'Batch Processing' button.

Set up your etomo session thusly:

Batch Setup Tab

- Select 'Move all stacks to dataset directory under:' and enter the path to your /tiltseries/etomo directory

Stacks Tab

- Add the stacks made in the last Warp step. The stacks will be .st files.

Dataset Values (Basic Tab)

- Set fiducial bead size in nm (my BSA-coated 5nm gold beads are closer to 8nm)
- 'Coarse aligned stack binning - single frame' should be *1* (no binning)
- 'Aligned stack binning' should be *1* (no binning)
- Select 'R-weighted back projection' and deselect 'SIRT-like filter'

Dataset Values (Advanced Tab)

- Set 'Tilt Axis Rotation' (mine was 85)
- 'Use existing .rawtlt file for A/only axis:' should be set to *Yes.* ***(IMPORTANT)***
- 'Type of magnification solution' should be *0: fixed*
- 'Type of rotation solution' should be *5: grouped*
- 'Type of beam tilt solution' should be *-1: solve one*

Run Tab

- Create the file /usr/local/ImodCalib/cpu.adoc **per these instructions**.

FILE cpu.adoc 0B

'number' is the number of logical cpu cores on your machine. 'gpu.device' is the list of GPUs on your machine (note the list starts at 1, unlike CUDA device nomenclature).

- With the .adoc created, select your desired *# CPUs* value (I chose 56).
- Select 'Parallel GPUs' and set your *# GPUs* value (I chose 2)

- Select 'Run multiple batch jobs in parallel' if you dare. I was able to run 7 jobs with a maximum of 2 GPUs per job.
- Under 'Subset of Steps to Run', check *Stop after* and select *Fine alignment.* ***(IMPORTANT)***
- Click *Run* and rock and roll

This will output two alignments files for each tiltseries - .xf and .tlt.

Warp

5 Head back to warp to import the etomo alignments and get ready for reconstruction. Still following **this guide.**

5.1 **Import alignments**
Copy the .xf and .tlt files from their etomo subdirectories and move to their own folder at tiltseries/imod_xf_tlt.

Then run

```
WarpTools ts_import_alignments --settings tiltseries.settings --
alignments tiltseries/imod_xf_tlt --alignment_angpix 6.5
```

Remember we made our stacks at 6.5Å in step 2.4 and did no binning in etomo, so the alignments were all calculated at 6.5Å.

5.2 **Check Handedness**
Back to **this tutorial**.

```
WarpTools ts_defocus_hand --settings tiltseries.settings --check
```

Follow the instructions in the output of this job to change your handedness if necessary.

Fidder

6 **Fidder** - involves two steps; creating masks of fiducials and erasing them. We have provided an example python script to complete both steps on your full and half micrographs so long as they are within the warp file structure. There is also the option to use the command-line for fidder.

6.1 You should only have to edit the parent_dir variable.

```
#!/usr/bin/env python3

import os
from multiprocessing import Pool
import mrcfile
import torch
from fidder.predict import predict_fiducial_mask
from fidder.erase import erase_masked_region

parent_dir = '/path/to/you/warp/frameseries/'

def check_directories(parent_dir):
    for subdir in subdirs:
        dir_path = os.path.join(parent_dir, subdir)
        if os.path.exists(dir_path):
            print(dir_path + ' found.')
        if not os.path.exists(dir_path):
            os.makedirs(dir_path)
            print(dir_path + ' created.')

    os.makedirs(os.path.join(parent_dir, 'average_erased'),
exist_ok=True)
    os.makedirs(os.path.join(parent_dir, 'average',
'even_erased'), exist_ok=True)
    os.makedirs(os.path.join(parent_dir, 'average', 'odd_erased'),
exist_ok=True)



def make_mask(filename, parent_dir):
    """ Apply fidder's predict_fidcucial_mask function to a single
micrograph and save the resultant mask as a new mrc file.

    Args:
        filename (str) : The name of the micrograph to process.
        parent_dir (str) : The directory containing micrographs in
/average and frame-split halves in
            /average/even, and /average/odd subdirectories.
    """
    mic_path = os.path.join(parent_dir, 'average', filename)
    mask_path = os.path.join(parent_dir, 'average', 'mask',
filename)

    image = torch.tensor(mrcfile.read(mic_path))
    mask, probabilities = predict_fiducial_mask(
        image, pixel_spacing=1.35, probability_threshold=0.7
    )

    mask_uint8 = mask.to(torch.uint8)

    os.makedirs(os.path.dirname(mask_path), exist_ok=True)

    with mrcfile.new(mask_path, overwrite=True) as mrc:
        mrc.set_data(mask_uint8.numpy())

    print(mask_path + '/' + filename + ' mask made.')

def erase_gold(filename, parent_dir, subdir):
    """Apply fidder's erase_masked_region function to a single
frame and save the result as a new mrc file.

    Args:
        filename (str): The name of the micrograph to process.
        parent_dir (str) : The directory containing micrographs in
/average and frame-split halves in
            /average/even, and /average/odd subdirectories.
    """
    mask_path = os.path.join(parent_dir, 'average', 'mask',
filename)

    mic_path  = os.path.join(parent_dir, subdir, filename)

    if subdir == 'average':
        output_subdir = os.path.join(parent_dir, 'average_erased')
    else:
        output_subdir = os.path.join(parent_dir, subdir +
'_erased')

    os.makedirs(output_subdir, exist_ok=True)

    image = torch.tensor(mrcfile.read(mic_path))
    mask = torch.tensor(mrcfile.read(mask_path))

    erased_image = erase_masked_region(image=image, mask=mask)

    mrc_output_path = os.path.join(output_subdir, filename)
    with mrcfile.new(mrc_output_path, overwrite=True) as mrc:
        mrc.set_data(erased_image.numpy())
    print(output_subdir +  '/' + filename + ' completed')

def process_gold(subdir, parent_dir):
    filenames = [i for i in os.listdir(os.path.join(parent_dir,
subdir)) if i.endswith('.mrc')]
    with Pool(48) as p:
        p.starmap(erase_gold, [(filename, parent_dir, subdir) for
filename in filenames])
    print('################################ all gold erased for '
+ subdir + ' ################################')


subdirs = ['average', 'average/even', 'average/odd']

check_directories(parent_dir)

filenames = [i for i in os.listdir(os.path.join(parent_dir,
'average')) if i.endswith('.mrc')]
for filename in filenames:
    mask_path = os.path.join(parent_dir, 'average','mask',
filename)
    if not os.path.exists(mask_path):
        make_mask(filename, parent_dir)
    else:
        print(f'{filename} aleady exists')
print('################################ mask processing complete
################################')

for subdir in subdirs:
    process_gold(subdir, parent_dir)
```

Depending on your computational resources, you may want to change the Pool() variable. erase_gold runs well with Pool(48) on our 64-thread machine.

Rename your existing (gold-containing) micrograph directories so they are not recognized by warp (e.g. /averages becomes /averages_gold etc.)

Move your newly gold-erased micrographs to the directories you just renamed (e.g. /frameseries/average, /frameseries/average/even, frameseries/average/odd)

To reconstruct your beautiful gold-erased tomograms:

```
WarpTools ts_reconstruct --settings tiltseries.settings --
dont_invert --halfmap_frames --angpix 10
```

Protocol references

Tegunov, D., Cramer, P. Real-time cryo-electron microscopy data preprocessing with Warp. *Nat Methods* **16**, 1146–1152 (2019). https://doi.org/10.1038/s41592-019-0580-y

Mastronarde DN, Held SR. Automated tilt series alignment and tomographic reconstruction in IMOD. J Struct Biol. 2017 Feb;197(2):102-113. doi: 10.1016/j.jsb.2016.07.011. Epub 2016 Jul 19. PMID: 27444392; PMCID: PMC5247408.

Kremer JR, Mastronarde DN, McIntosh JR. Computer visualization of three-dimensional image data using IMOD. J Struct Biol. 1996 Jan-Feb;116(1):71-6. doi: 10.1006/jsbi.1996.0013. PMID: 8742726.