Code Description

MATLAB scripts in the folder "OTF_simulation"

We provide 2 main MATLAB scripts for simulations of 2D and 3D OTF supports that run on MATLAB version 2016b and above.

1. **otf_2D_filled.m**: simulations of 2D OTF support (xz profile) of widefield microscopy, I^2M, standing-wave microscopy, 3D SIM, 4-beam SIM and I^5S.
2. **otf_3D_rendering_fsurf.m**: simulations of 3D OTF support of 4-beam SIM.

MATLAB scripts in the folder "SLM"

We provide 1 main MATLAB script for SLM pattern generation that runs on MATLAB version 2016b and above.

1. **SLM_pattern.m**: binary phase grating pattern generation with i) the desired pattern orientations, ii) the desired line spacing (grating period) in each pattern and iii) the desired duty cycle.

Key function(s):

```
function SLM = patterntabArb(vecA, period, onfrac, phaseInd, phaseOffset,
nphases, sizex, sizey)
```

a) "vecA" specifies pattern orientation. In this work, we chose vecA = (2,-11), (14,-5), and (13,11) as the three pattern orientations, which correspond to 10.3°, 70.3° and 130.2°.
b) "period" defines the grating period.
c) "onfrac" specifies the duty cycle of the SLM pattern, defined by the fraction of pixels in the on-state (π phase retardance) in each period.
d) "phaseInd" defines the phase index corresponding to all the phases of each pattern orientation. It is within [0, 14] for 3D SIM and [0, 8] for 2D SIM.
e) "phaseOffset" defines the phase offset added to all the phases, usually defaults to 0.
f) "n phases" defines how many phases within each pattern orientation. It is 5 for 3D SIM and 3 for 2D SIM.
g) "sizex" is the horizontal pixel size of the SLM active area, e.g., 1920.
h) "sizey" is the vertical pixel size of the SLM active area. e.g., 1152.

MATLAB scripts in the folder "OTF"

We provide 1 main MATLAB script for 3D SIM & 4-beam SIM OTF generation that runs on MATLAB version 2016b and above.

1. **makeotf.m**: Produces a radially averaged OTF starting from an experimental PSF dataset. Used mainly for structured illumination microscopy (3D SIM or 4-beam SIM). It also can: (1) compensate for finite bead size; (2) incorporate the phase factor associated with the side bands;

(3) estimate the sub-pixel position of the bead by fitting parabolas; (4) clean up out-of-band noise.

There are 2 accessory scripts that are called after running "**makeotf.m**" and that must be used together:

1. **ImageJ_formatted_TIFF.m**: A script that writes and reads tiff files in ImageJ format with metadata.
2. **Simple_IFD.m**: A script that defines a simple tiff image header called "IFD" (Image File Directory).

There are also 8 PSF datasets for testing, corresponding to PSFs in both 3D SIM and 4-beam SIM modes acquired with 1.27 and 1.35 NA objectives at 488nm and 561nm, respectively, named:

1. "PSF_3DSIM_1P27NA_488.tif"
2. "PSF_3DSIM_1P27NA_561.tif"
3. "PSF_3DSIM_1P35NA_488.tif"
4. "PSF_3DSIM_1P35NA_561.tif"
5. "PSF_4BSIM_1P27NA_488.tif"
6. "PSF_4BSIM_1P27NA_561.tif"
7. "PSF_4BSIM_1P35NA_488.tif"
8. "PSF_4BSIM_1P35NA_561.tif"

Before running the code, in Lines 15-19, users need to decide which PSF to process. For example, when choosing `NA = 1.27; lambda = 488; twolens = false;` "PSF_3DSIM_1P27NA_488.tif" will be processed and the corresponding OTF will also be generated.

OTF data (complex single MATLAB matrix) and OTF image (32-bit TIFF stack) will be saved in the same folder as "OTF_3DSIM_1P27NA_488.mat" and "OTF_3DSIM_1P27NA_488.tif", respectively. The OTF image contains metadata including pixel width, pixel height and voxel depth, so it must be used together with the OTF data. "OTF_exp.tif" is the OTF image after raising the contrast for display purposes. "bands.tif" is the intermediate OTF result.

MATLAB scripts in the folder "Sirecon"

We provide 1 main MATLAB script for 3D SIM & 4-beam SIM Wiener reconstruction that run on MATLAB version 2016b and above.

1. **SireconDriver.m**: all calculation starts with a "SIM_Reconstructor" object in the following steps: (1) Instantiate a "SIM_Reconstructor" object by passing along the command line; (2) For all time points, do: (2a). Load raw data and preprocess; (2b). call member function processOneVolume() to reconstruct the current time point; (2c). write reconstruction result of current time point; (3) Close file(s) and exit.

There are 4 accessory scripts that are called after running the main software "**SireconDriver.m**":

1. **ImageJ_formatted_TIFF.m**: A script that writes and reads tiff files in ImageJ format with metadata. It must be used together with 2.

2. **Simple_IFD.m**: A script that defines a simple tiff image header called “IFD” (Image File Directory). It must be used together with 1.
3. **SIM_Reconstructor.m**: A script that defines a class which contains all the basic functions to process SIM images.
4. **OtfProvider.m**: A script that defines a class which contains all the basic functions to process OTF data.

   Before running the code, in Lines 19-79, users need to specify the following parameters:

   1) `'ifiles'`: defines the raw SIM data file path, e.g., 'D:\SIMreconProject\Sirecon\DataForTest'. More test datasets can be downloaded from Zenodo.
   2) `'ofiles'`: defines the raw SIM data filename (or part of the filename), e.g., 'bac_3DSIM(1P35NA)'. For time-series data, they should be named as 'bac_3DSIM(1P35NA)_00', 'bac_3DSIM(1P35NA)_01', 'bac_3DSIM(1P35NA)_02', …. etc., where the number indicates the relevant time point.
   3) `'otffiles'`: defines the OTF filename, e.g., 'D:\SIMreconProject\Sirecon\OTF\OTF_3DSIM_1P35NA_488.tif'. Note: “OTF_3DSIM_1P27NA_488.mat” and “OTF_3DSIM_1P27NA_488.tif” must be located in the same folder. The correct OTF file should be chosen, i.e., the imaging mode (3D SIM or 4-beam SIM), NA (1.27 or 1.35), excitation wavelength (488 nm or 561 nm) and central emission wavelength (525 nm or 607 nm) must be consistent with the raw SIM data.
   4) `'usecorr'`: defines whether to use a flat-field correction file provided by the camera manufacturer. No assignment by default.
   5) `'ndirs'`: defines the number of orientations used in SIM acquisition, defaults to 3.
   6) `'nphases'`: defines the number of phases per orientation used in SIM acquisition, defaults to 5 in 3D SIM and 4-beam SIM.
   7) `'nordersout'`: defines the number of output orders, defaults to 0.
   8) `'angle0'`: defines the angle of the first orientation in radians. It defaults to -0.19142 for the 1.35 NA setup, and -1.3994 for the 1.27 NA setup.
   9) `'ls'`: defines the line spacing(s) of the SIM pattern in microns in the image plane. Default values are given in the commented section below `'wavelength'`
   10) `'na'`: defines the detection numerical aperture of the objective (1.27 or 1.35).
   11) `'nimm'`: defines the refractive index of the objective immersion medium (1.406 for silicone oil or 1.33 for water).
   12) `'nmed'`: defines the refractive index of the sample medium (1.406 for 45.6% Iodixanol solution or 1.33 for water / PBS).
   13) `'iter_input'`: defines the iteration number for input Richard-Lucy (RL) deconvolution. All raw SIM images (input) will first be 2D deconvolved using the RL algorithm with this iteration number before participating in subsequent computations. We do not use this parameter for this work.
   14) `'iter_output'`: defines the iteration number for output RL deconvolution. The widefield image (output #1) will be 2D / 3D deconvolved with this iteration number. Another super-resolution image (output #2) will be computed based on 2D / 3D deconvolution algorithm

with the same iteration number other than Wiener reconstruction for comparison purposes. We do not use this parameter for this work.

15) `'zoomfact'`: defines the lateral zoom factor after Wiener reconstruction, defaults to 2, meaning that the lateral pixel number will be increased by a factor of 2 after reconstruction.
16) `'explodefact'`: code will artificially expand the reciprocal-space distance between orders by this factor. It defaults to 1.
17) `'zzoom'`: defines the axial zoom factor, defaults to 1.
18) `'background'`: defines the camera readout background, which will be subtracted from input images, defaults to 100 counts for our pco.edge 4.2HQ sCMOS camera.
19) `'wiener'`: defines the Wiener constant, defaults to 0.001.
20) `'linear_wiener'`: defines the Wiener constant increment between each time point. No assignment by default.
21) `'forcemodamp'`: forces the modulation depth to these values. No assignment by default.
22) `'k0angles'`: defines the angles of all 3 orientations in radians if we have prior knowledge of the system. Default values were given in the commented section below `'wavelength'`.
23) `'gammaApo'`: defines the output apodization gamma, defaults to 1 (triangular shape).
24) `'saveprefiltered'`: defines whether to save separated bands (in the frequency domain) into a file and exit. Only used for testing. No assignment by default.
25) `'savealignedraw'`: defines whether to save drift-corrected raw data (half Fourier space) into a file and exit. Only used for testing. No drift correction within three orientations is needed for our system. No assignment by default.
26) `'saveoverlaps'`: defines whether to save overlap0 and overlap1 (real-space complex data) into a file and exit. Only used for testing. No assignment by default.
27) `'config'`: defines whether to save a configuration file.
28) `'lambda'`: defines the excitation wavelength in nanometers (488 nm or 561 nm).
29) `'wavelength'`: defines the central emission wavelength in nanometers (525 nm or 607 nm).
30) `'useRLonInput'`: defines whether to use 2D RL deconvolution on raw input images, not used by default and commented out.
31) `'useRLonOutput'`: defines whether to use 2D / 3D RL deconvolution on output images, not used by default and commented out.
32) `'nofilteroverlaps'`: if true, does not filter the overlapping region between bands, usually used in trouble shooting. We choose to filter the overlapping region and comment this parameter out.
33) `'otfRA'`: defines whether to use a rotationally averaged OTF, defaults to yes.
34) `'otfPerAngle'`: defines whether to use the same OTF for all 3 orientations, defaults to yes.
35) `'fastSI'`: defines raw SIM organization in XYPAZ format, defaults to yes. Otherwise, data are saved as in OMX format (XYPZA). P: phase; A: orientation; Z: depth.
36) `'searchforvector'`: defines whether to search for more accurate pattern vector k0 at the 0th time point. If the SNR of the raw SIM data is very low, a warning message will appear, meaning that the k0 search algorithm fails. In this case, comment this parameter out.

37) `k0searchAll'`: defines whether to search for k0 at all time points, defaults to no. Used for processing time-series only.
38) `'equalizez'`: defines whether to do bleach correction between different depths, defaults to yes.
39) `'equalizet'`: defines whether to do bleach correction between different time points, defaults to yes.
40) `'dampenOrder0'`: defines whether to dampen order-0 in the final Wiener reconstruction, defaults to yes.
41) `'nosuppress'`: if used, does not suppress DC singularity in the final Wiener reconstruction. Defaults to no.
42) `'nokz0'`: if used, does not use kz=0 plane of the 0th order in modulation amplitude fitting and function assemblerealspace(). We always choose to use the kz=0 plane of the 0th order and comment this parameter out.
43) `'twolenses'`: defines whether the raw SIM data is acquired in 3D SIM or 4-beam SIM mode.
44) `'applyOtfBeforeShift'`: decides whether to multiply bands with OTF before frequency shifting or after frequency shifting. The difference in output between the two choices is minor.
45) `'keepnegativevalues'`: decides whether to keep negative values when saving Wiener reconstructions, defaults to no.
46) `'savefft'`: decides whether to save fft data, defaults to no.
47) `'fftkeepnegativevalues'`: decides whether to use negative values when saving fft data, defaults to yes.
48) `'GPUcomputing'`: decides whether to use GPU acceleration, defaults to no.

After running the software, a widefield image ("wide_field_0.tif"), a Wiener reconstructed image ("Wiener_0.tif") and 3 single-orientation Wiener reconstructed images ("Wiener_dir_1_0.tif", "Wiener_dir_2_0.tif" and "Wiener_dir_3_0.tif") are saved under the same folder as the raw SIM data (`'ifiles'`). Similarly, for time-series data, a series of widefield images ("wide_field_0.tif", "wide_field_1.tif", "wide_field_2.tif", …, etc.) and Wiener reconstructed images ("Wiener_0.tif", "Wiener_1.tif", "Wiener_2.tif", …, etc.) will be saved, where the number indicates the relevant time point.

Python scripts in the folder "DeepLearning" for denoising

The code for denoising 3D SIM images was based on 3D RCAN. Both training and prediction scripts (train.py and apply.py) can be downloaded from https://github.com/AiviaCommunity/3D-RCAN. Users should follow the instructions listed in requirements.txt. Here we provide a test dataset (15 raw SIM images, named lowSNR_1 to lowSNR_15) and an RCAN model (EMTB_DenoisingModel_Step1) for the 1st denoising step. Users can then run the MATLAB code to run 3D SIM reconstruction (i.e., process the denoised raw data for 3D SIM), and use another RCAN model (EMTB_DenoisingModel_Step2) for the 2nd denoising step. Test datasets can be downloaded from Zenodo.

Python code for improving z resolution

1. The code IsotropicZ_Github.py was developed for improving axial resolution in 3D SIM. This script was written based on using CARE predictions. Before using it, users need to download CARE, taking care to observe the associated requirements (e.g., TensorFlow and Keras) from https://github.com/CSBDeep/CSBDeep
2. Users need to install Scipy (v. 1.6 or above): pip install scipy
3. In the script, in Lines 21 and 22, users need to choose 'training' or 'prediction' mode.

```
#mode ='training'
mode='prediction'
```

4. If setting 'training' mode, then in Lines 25-27, users need to define the training directory, 'input' folder, and 'ground truth' folder.

```
training_dir = 'Z:\\3D_SIM\\xxxx'
input_folders = ['XZ_1DSIM_Input']
truth_folder  = 'XZ_1DSIM_GT'
```

5. If setting 'prediction' mode, then in Line 34, users need to define the prediction directory that includes tiff images for deep learning recovery. Prediction results will be automatically saved under a new folder "prediction_dir + CARE_6degreee_DL".

```
prediction_dir = 'Z:\\3D_SIM\\xxxx'
```

6. Either with 'training' or 'prediction' mode, in Lines 31 and 32, users need to set a directory and a name to save the trained model or call the model for application.

```
model_dir ='Z:\\3D_SIM\\xxxx'
model_name = 'CARE_XZ_6degree_Model_Lamp1'
```

7. In Lines 49-50, users need to set the pixel size for the x, y and z dimensions of the raw data.
8. In Lines 56-57, uses need to set sigma values (along x dimension and z dimension) of the blurring kernel so that after blurring, the frequency spectrum of XZ slices look symmetric.
9. In Line 63, users can set the value of 'save_flag' to 1 (i.e., saving all intermediate images including the deep learning results of each rotation angle), or to 0 (i.e., saving only the final results, i.e., combination from the 6-rotation predictions).
10. Note that during the CARE training, the patch size must be a multiple of 4. In Lines 77-91, the patch size is set to 64 or the pixel size of the input data if it is smaller than 64. In the latter case, users need to crop the data so that the pixel size is a multiple of 4.
11. In Line 121, the training epochs is set to 100, and steps per epoch is 100. Users can increase or decrease the number to double check deep learning performance.

```
config = Config(axes, n_channel_in, n_channel_out, train_epochs = 100, train_steps_per_epoch=100)
```

12. In Line 140, the rotation angle is set at -90, -60, -30, 0, 30, 60 degrees. Users can define other angles.

```
angle = np.linspace(-90, 90, 7)
```

13. Here we provide a CARE model (EMTB_IsotropicZModel_Step3) to apply after the 2nd step of denoising described above. Users can compare the result with our result denoising and axial resolution enhancement - 'Final_DL_Result.tif'.