USER MANUAL OF F-VCD PACKAGE

Abstract

Detailed data processing and network training & validation of F-VCD

Chengqiang YI

cqyi@hust.edu.cn

Contents

1. **Training pairs generation** 2
   - A. PSF simulation. 2
   - B. Rescaling ground truth data. 2
   - C. Light field projection. 3
   - D. Adding noise. 3
   - E. Crop training pairs 4
2. **F-VCD Training && Validation** 4
   - A. Environment installation. 4
   - B. Network Training. 5
   - C. Validation 5
3. **Ref**. 5

1. Training pairs generation

A. PSF simulation.

Note: Before PSF simulation, users need to download Matlab first.

We use wave-optics-based simulation code to generate designed FLFM PSF. This PSF simulation code is modified from olaf[1], which is located at the sub folder '*./ Code/SimuProjection/*' as shown in Fig 1.

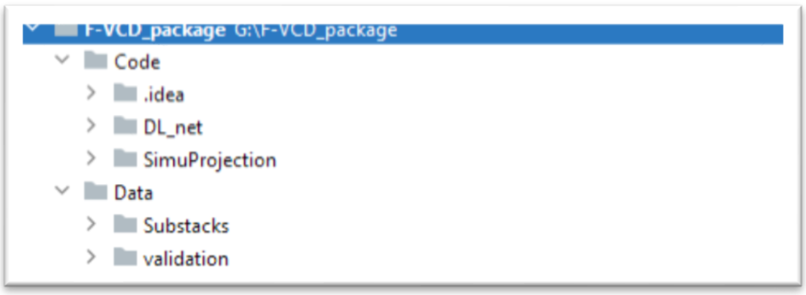


Fig. 1 Directory structure of F-VCD package

i. Users need to set their customized parameters in '*main_PSF_simulation.m*'.

For example, we use the following optics design:

Objective: 100x/1.4	---- > config.M=100; config.NA=1.4
Tube lens: Thorlabs TTL180	---- > config.f1=180000;
Fourier lens: ACT508-300	---- > config.f2=300000;
MLA (f=120mm, pitch=3250μm)	---- > config. lensPitch=3250; Config.fm=120000;
sCMOS(pixel size=6.5μm)	---- > config.spacingPixel=3250/6.5=499;

Note: 'config.SensorSize' only need to be a little larger than the size of back pupil for fast calculation. For example, it set as 1497 pixels while pupil size is 1.4/100*2*300000/6.5=1292.

ii. After calculation, the PSF matrixes are automatically saved in the corresponding folder, e.g. 'View3_simu_[m3.4-3.4]_zstep_0.17' which contains '*H.mat*' for forward projection and '*Ht.m*' for deconvolution as shown in Fig. 2.

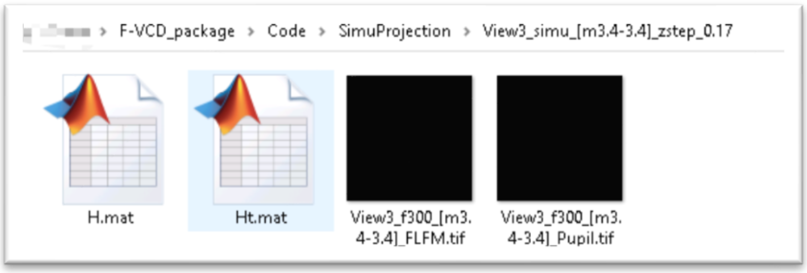

F-VCD_package > Code > SimuProjection > View3_simu_[m3.4-3.4]_zstep_0.17
H.mat
Ht.mat
View3_f300_[m3.4-3.4]_FLFM.tif
View3_f300_[m3.4-3.4]_Pupil.tif

Fig. 2 Simulated light-field PSF

B. Rescaling ground truth data.

Decide the super-resolution factor of F-VCD and rescale raw data to the specific 'voxel size'. For example, the voxel size of simulated PSF of our designed triple-views FLFM system are

0.1625μm at lateral dimension and 0.17 μm at axial dimension, which is about three-folds than the high-resolution stack acquired from Airyscan2. So, we set the SR factor as 3 and scale the acquired stacks to 0.1625/3≈0.054 μm at XY dimension and 0.17 μm at Z dimension as shown in Fig.3.

C. Light field projection.

i. Down-sample the rescaled high-resolution data according to the SR factor

ii. Click '*ForwardBatch_GPU.m*' and set the following parameters:
   - GPU_enable: whether to use GPU
   - desired_depth: the number of slices of input stack.
   - psf_path: the name of folder which contains PSF matrixes

iii. Run '*toViewStack.m*' which is used to extra views from LF projection.

D. Adding noise.

We add Gaussian and Poisson noise to the generated 'clean LF' according to the SNR measurement of experimental data. Users can measure the deviation and mean value of corner ROI of their own LF acquisition to derive the 'std' parameter of noise and background value respectively.

Note: There may exist fluorescent background in captured FLFM. Users can add matched offset or defocus signals to mimic this background.

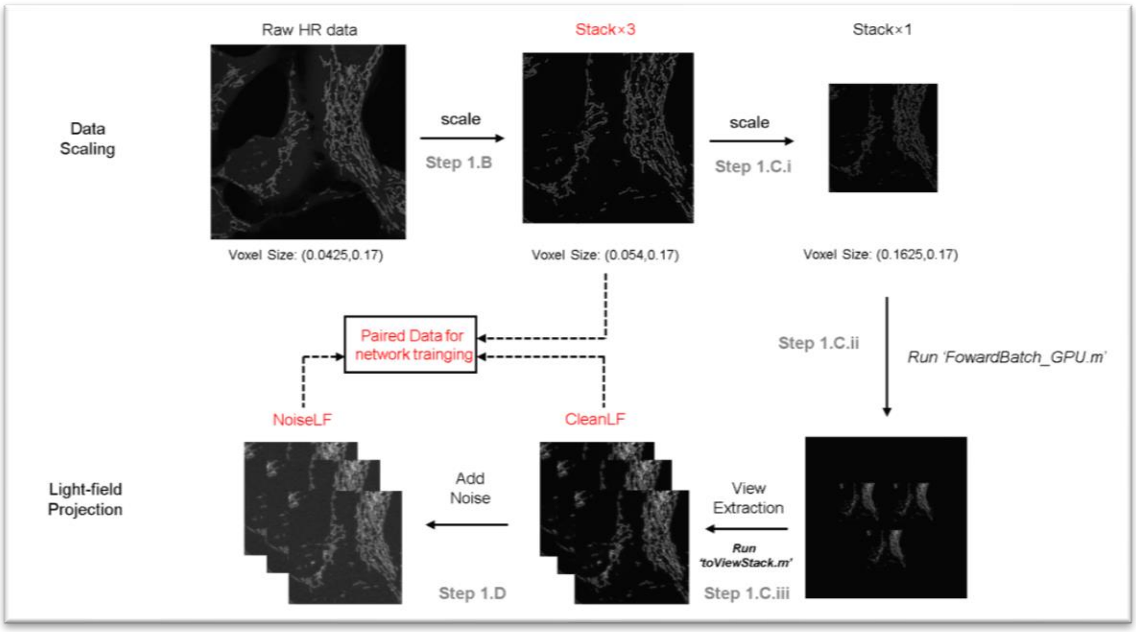


Fig. 3 Pipeline of data-preprocessing.

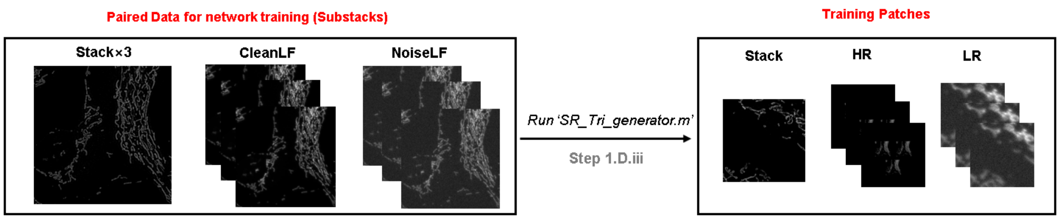


Fig. 4 Training Pairs Cropping.

E. Crop training pairs (Fig. 4)

a. Click '*SR_Tri_generator.m*' to generate training patches. Users need to set the following parameters:

- lr//hr//GT_path: data path of noiseLF // cleanLF// GT_path
- LF_cropped_size: the lateral size of cropped 'clean LF'
- SR_factor: upsampling factor from 'Noise LF' to 'Clean LF'. Default is 1.
- rescale_factor: rescaling factor from 'Stack to 'Clean LF'. Default is 1/3.
- pixel_ratio: the threshold to decide whether to save cropped stacks. Default is 0.7. Larger ratio means less data.

Note: For convenience, we have uploaded rescaled stacks and corresponding LF projection at Google drive, including:

Substacks.zip:

- Stack×3: scaled HR data as Groundtruth
- Stack×1: scaled stacks for LF projection
- CleanLF: view stacks without noise
- NoiseLF: view stacks with noise and background

Mito_View3_[LF160_SF160_ Stack480].zip:

- Mito_View3_[LF160_SF160_Stack480]: cropped training pairs.

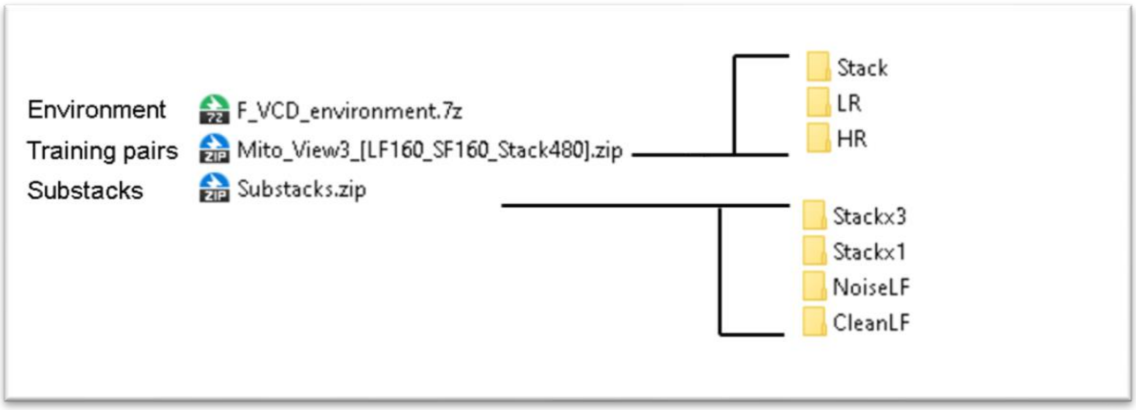


Fig. 5 Uploaded data

2. F-VCD Training && Validation

The source code of F-VCD is located at './Code/ DL_net'. This step explains how to install required packages and train F-VCD network.

A. Environment installation.

We provide a '. yml' file (at './Code/DL_net/environment') which includes all packages' name and version for quick installation. Typing one command in Anaconda Prompt:

```
> conda env create -f vcd_environment.yml
```

Critical: F-VCD network is written by Tensorflow 1.x. If using RTX30 or RXT40 series GPU, users need to install specific version of Tensorflow-GPU for Ampere GPU, which can be downloaded from:

https://github.com/fo40225/tensorflow-windows-wheel/tree/master/1.15.4%2Bnv20.12/py38/CPU%2BGPU/cuda111cudnn8avx2

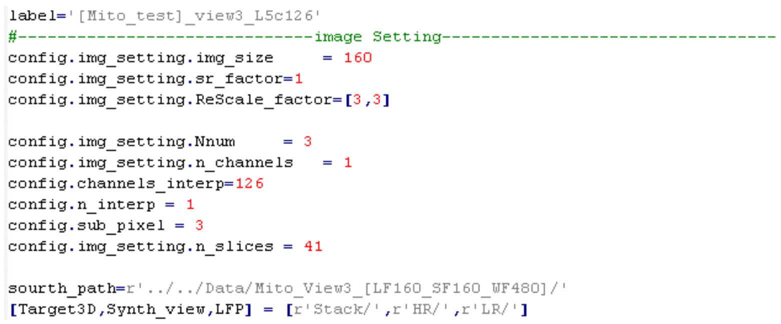
Besides, we also provided the compressed file of virtual environment of Anaconda (users can download from Google drive). Users can directly unzip this file into their own environment folder (e.g. './anaconda3/envs/). Meanwhile, users need to install corresponding *Cuda* and *cuddn*. We test this zipped environment on Cuda v11.1 and cuddn v8.2.0.

B. Network Training.

a. Parameter settings.

Before network training, some hyperparameters and image information need to be entered into 'config.py'.

- label: the name of training model
- config.img_setting.img_size: patch size of input noisy LF.
- config.img_setting.sr_factor: upsampling factor from noisy LF to clean LF
- config.img_setting.ReScale_factor: upsampling factor from clean LF to 3D stacks
- config.img_setting.Nnum: the view number of input noisy LF
- config.channels_interp: filter number of F-VCD
- config.sub_pixel: upsampling factor from clean LF to 3D stacks
- sourth_path: the path of cropped training pairs



```
label='[Mito_test]_view3_L5c126'
#------------------------------image Setting----------------------------------
config.img_setting.img_size     = 160
config.img_setting.sr_factor=1
config.img_setting.ReScale_factor=[3,3]

config.img_setting.Nnum     = 3
config.img_setting.n_channels   = 1
config.channels_interp=126
config.n_interp = 1
config.sub_pixel = 3
config.img_setting.n_slices = 41

sourth_path=r'../../Data/Mito_View3_[LF160_SF160_WF480]/'
[Target3D,Synth_view,LFP] = [r'Stack/',r'HR/',r'LR/']
```

Fig. 6 Screenshot of network configuration settings

b. Model training.

After entering the basic parameters, users can run 'train.py' to train F-VCD network.

Note: We recommend using GPU with ≥24GB memory.

C. Validation

When model training finished, the checkpoint would be automatically saved at '*./Code/DL_net/checkpoint/{defined label name}*'. To using this trained model to inference 3D stack from LF, users need to enter the validation data path in 'config.py' ('*config.VALID.lf2d_path*' at line 76) and run 'eval_test.py'

Note: We also uploaded the validation results at the folder './example/validation'

3. Ref.

1. Stefanoiu, A., Page, J. & Lasser, T. "olaf: A flexible 3d reconstruction framework for light field microscopy". (2019).