# Code Package for Dvem-TPM

**PSF generation Procedure**

As an example, the point spread function (PSF) of Dvge-TPM was generated by spatially multiplexing 25 densely spaced Gaussian foci. Phase encoding data for the two excitation arms were stored in GradF\_phase\_1 and GradF\_phase\_2, each containing three components:

* GradF\_phase\_X{1} stores the Zernike defocus coefficients
* GradF\_phase\_X{2} contains phase modulator added to modulate optical interference among the foci
* GradF\_phase\_X{3} provides binary matrix corresponding to each focus

To synthesize the phase pattern loaded on the spatial light modulator (SLM), the function setZernike\_nmcoeff (from class FourierOptics.m) was used to generate the defocus phase term corresponding to each Gaussian focus. The phase modulator was subtracted, and then a binary matrix was applied to each focus. The resulting terms were accumulated to form the final phase pattern phase\_sum\_1 and phase\_sum\_2 for the two arms.

Light field simulations were performed using the function getLightField (from class FourierOptics.m), where the pupil function defined by the NA of the system. The simulated intensity distributions I1 and I2 present the axial profile of the synthesized elongated PSF.

**Depth decoding Procedure**

As a demonstration dataset, we used calcium imaging data from the primary motor cortex (M1) and primary somatosensory cortex (S1) of an awake mouse (a dataset also presented in the main text). The data (demo\_L.tif and demo\_R.tif) were preprocessed to remove rigid lateral motion artifacts using the TurboReg plugin in Fiji.

To identify and segment active neurons, the dataset was processed using the CaImAn by running the following commands:

>> nam =[' demo\_L.tif'];

run(fullfile(pwd, 'demo\CaImAn-MATLAB-master', 'demo\_script\_main.m'));

nam =[' demo\_R.tif'];

run(fullfile(pwd, 'demo\CaImAn-MATLAB-master', 'demo\_script\_main.m'));

To refine the ROIs (regions of interest), we filtered out poorly segmented neurons based on intensity, size, and shape (ellipticity) using:

>> [c\_L,NeurTT\_L] = preprocess\_components\_L(mask\_cell,C\_or,15,80,Df);

[c\_R,NeurTT\_R] = preprocess\_components\_R(mask\_cell,C\_or,5,40,Df);

*Note: The parameters in preprocess\_components\_L and preprocess\_components\_R should be fine-tuned based on the signal-to-noise ratio of the images and experimental conditions.*

Next, we loaded the PSF parameters required for depth decoding:

>> load('psf\_parameters\_test.mat');

We then computed the one-to-one correspondences between the projections in Im\_L and Im\_R using:

>> [final\_combinations,NeurTT]=proj\_pairs(NeurTT\_L,NeurTT\_R,c\_L,c\_R,Im\_L,Im\_R,distance,psf\_ratio\_new,kernel\_1,kernel\_2);

*The sixth column of final\_combinations contains the estimated depths of all active neurons, and NeurTT contains their corresponding calcium transients.*