Automating 3D Live-Cell Membrane Segmentation Through Deep Learning With 3D U-Net

Zeeshan Patel[1], Xiongtao Ruan[1], Cyna Shirazinejad[1], Shawn Hua[1], Aaron Sun[1]
Tian-Ming Fu[2], Dan Milkie[2], Wesley Legant[1,2], Eric Betzig[1,2], Srigokul Upadhyayula[1,2]
[1]University of California, Berkeley, Berkeley CA, USA
[2]Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, VA 20147, USA
[3]University of North Carolina, Chapel Hill, NC, USA

Abstract. Adaptive Optical Lattice light-sheet microscopy (AO-LLSM) is a 3D live-cell imaging method that reduces damage to the physiological state of the biological specimen due to phototoxicity while also maintaining high-resolution image quality by avoiding issues with aberrations (Gao et al., 2019; Liu et al., 2018). AO-LLSM delivers images in high spatiotemporal resolution, which allows for detailed analysis of individual cells in complex, multicellular organisms. However, identifying cell boundaries in dense, multicellular organisms can be a tedious task. In order to understand cellular processes, it is vital to properly identify and segment individual cell membranes. Precise labeling will enable the isolation of any cell in images of dense, multicellular organisms, allowing researchers to analyze their cellular dynamics, cell morphologies, and organelle processes. We outline a 3D-live cell membrane segmentation method using deep learning with 3D U-Net. We developed our own image normalization and ground truth label binarization algorithms for data preprocessing using core frameworks such as NumPy and scikit-image. To generate training and testing datasets, we created random augmentation algorithms that utilized Gaussian noise functions and TensorFlow based image augmentation functions. To develop the 3D U-Net, we utilized the Tensorflow-based Keras API and the original neural network architecture (Özgün Çiçek et al., 2016). To use our tool on large, volumetric datasets, we developed a prediction script that can take in 3D cell membrane images of any size and produce a predicted label for the image. Our 3D U-Net was able to generate segmentation labels in less than five minutes, with accuracies around 96.55%. The overall process allows for fast image processing and rapid neural network training, providing both a time and cost-efficient, automated method for 3D cell membrane detection and segmentation.

3D Cell Membrane Imaging of Zebrafish Embryo

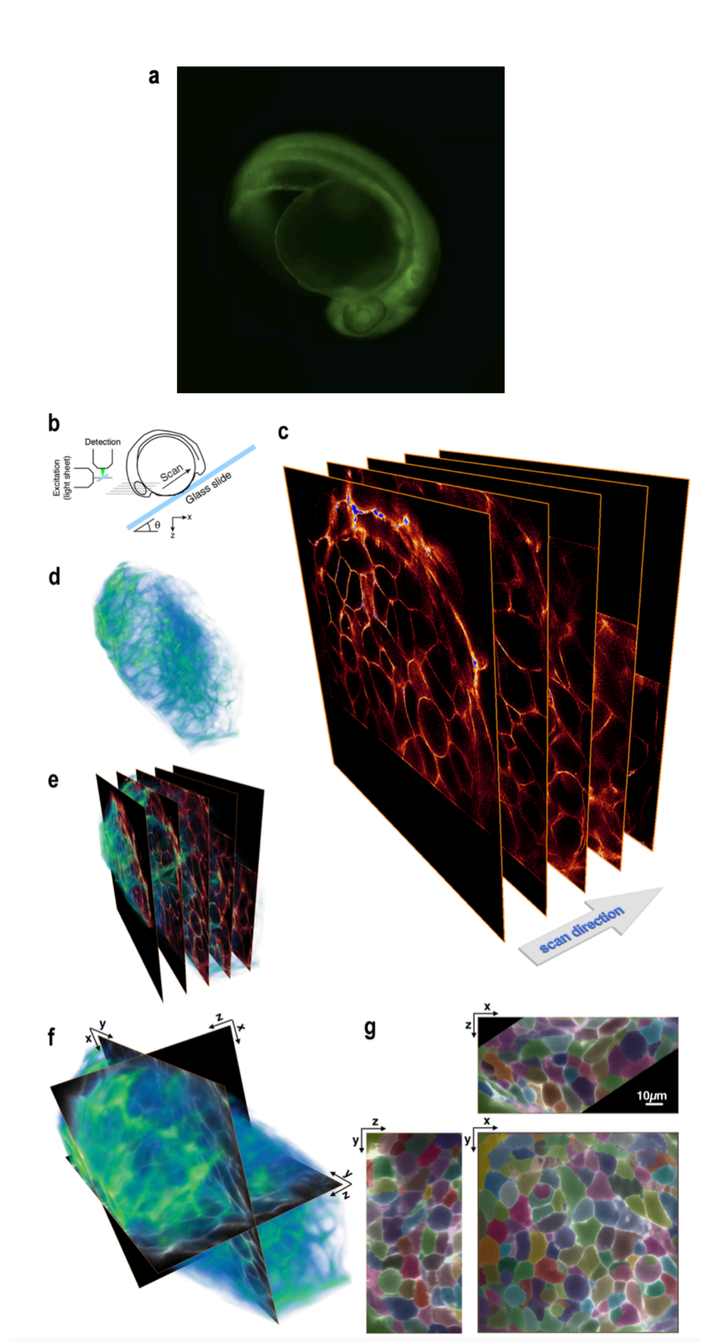

Figure 1. 3D schematic of the developing zebrafish embryo using AO-LLSM. **(A)** Image displaying the complete zebrafish embryo used mounted on the AO-LLSM. **(B)** Schematic of the orientation of the zebrafish embryo mounted on the AO-LLSM and the optical system. **(C-G)** Example in **(C)** shows specific areas of the zebrafish embryo eye obtained 17 hours after fertilization. Sections from the imaged volume (~80μm x 80μm x 100μm) displayed in **(D)**. Every image stack was obtained in 10 s succeeded by a 6 s pause and consisted of 251 planes acquired by scanning the embryo at 400 nm intervals. Deconvolved planes (red/yellow; b, d) of the tissue volume (green/blue; c, d) are shown every 50 slices, about 10μm. The volume (c, d) is at 50% transparency. **(F)** Graphic underscoring the relationship between the complete 3D image corresponding to a non-deconvolved point of the time series and orthogonal optical sectors located near the center of the imaged volume. **(G)** 3D segmentation output of the cell membranes which is used for the ground truth label inputted into the 3D U-Net. Cells randomly colored to represent membrane delineation.

Once the data is acquired, we go through every slice in the volume and remove slices with any artifacts to ensure that cleaned data is used for model training. This is done by overlaying the ground truth label on the raw cell membrane data and deleting slices with artifacts.

Image Normalization and Ground Truth Binarization

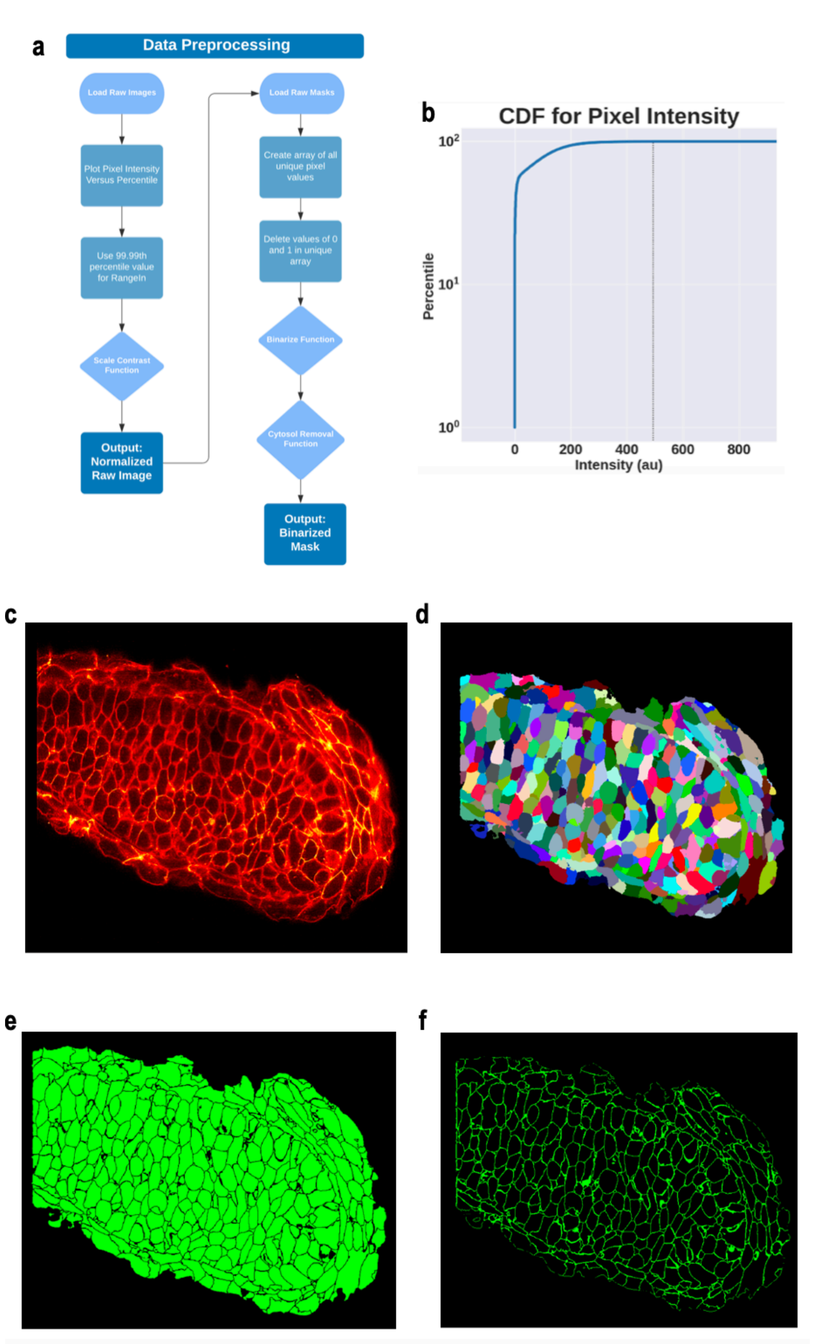

Figure 2. Preparing Raw 3D Cell Membrane Data Acquired from AO-LLSM for 3D U-Net Training. (A) Flow chart outlining step-by-step process for data preparation algorithm. Details the Image Normalization and the raw mask binarization methodology. Prior to data preparation, we performed data cleansing by removing slices from the original raw image and mask that contained artifacts. **(B)** Using a pixel intensity versus percentile chart, the user can specify the input pixel intensity range of the image. Our contrast scaling function then returns an 8-bit scaled raw image, ensuring that the user is not losing data. In **(B)**, we plot the pixel intensity versus percentile plot (log scale) for our sample image and determine the 99.99th percentile pixel value to be approximately 493 (dashed line), and we set RangeIn to (0, 493). **(C)** shows the normalized image while **(D)** displays the original mask. Overall, the contrast in the normalized image improves and it displays all of the data clearly. Afterwards, the corresponding mask for the raw image goes through the binarization function. First, an array of all unique values in the mask is initialized, and values of 0 and 1 are removed. This array is then inputted into the binarization function, which returns the binary label **(E)**. This binary label goes through another function to remove all cytosolic values in the binary mask, leaving only the cell membranes as seen in **(F)**.

Once both raw images and binary masks are prepared, they can be put through our random augmentation function, which will create a crop and randomly augment it using Tensorflow image flipping and rotation functions. Additionally, the noise can also be added to the images to diversify the signal-to-noise ratio in the training and testing datasets. The noise function is based on a user-specified mean (μ) and standard deviation (σ) based on a Gaussian distribution. We also developed a noise function in which the user can choose noise levels with pre-determined Gaussian means and standard deviations that worked well for our datasets. These functions are all called in a loop to generate multiple images, which are saved using the tifffile Python library.

3D U-Net

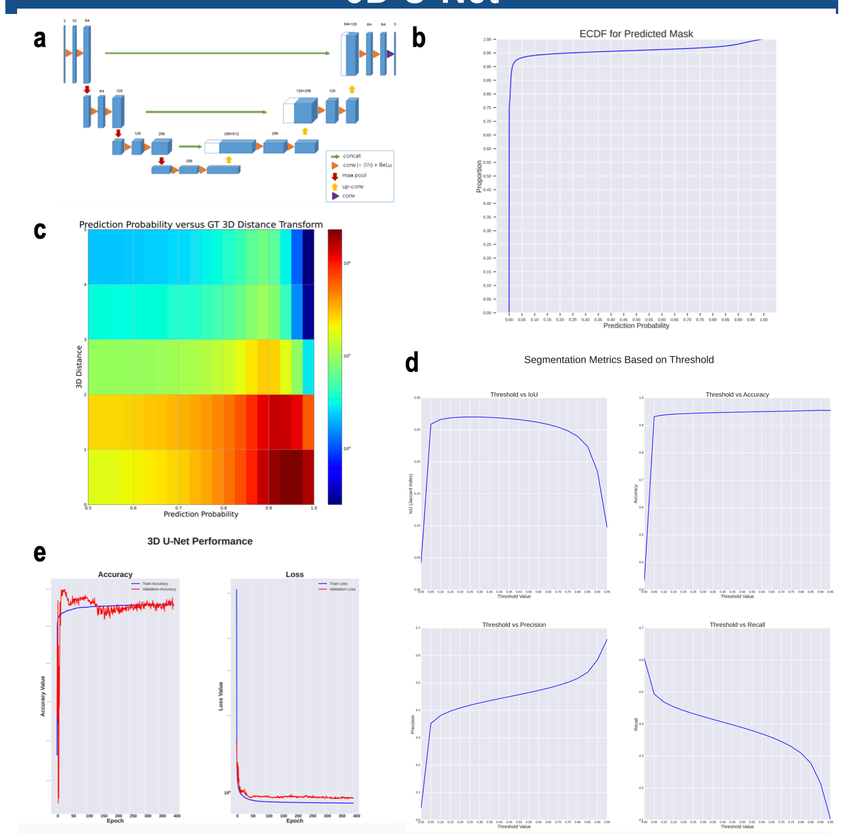

Figure 3. 3D U-Net Architecture and Training Performance. (A-E) The 3D U-Net architecture used for our segmentation tool is shown along with the neural network performance during four trainings. **(A)** The schematic for the 3D U-Net framework displays two main paths: the Contraction Path and Expansive path. The Contraction Path consists of groups of 3D Convolution layers, Batch Normalization layers, and 3D Max Pooling layers. The Expansive Path is composed of groups of 3D Convolution Layers, 3D Convolution Transpose layers, Concatenation layers, and Batch Normalization layers. All of the code for each layer is derived from the Tensorflow-based Keras API. **(B-E)** Best 3D U-Net training which ran for 375 epochs with a training dataset of 6,600 images, out of which a randomly selected set of 1,280 images were utilized for validation. Trainings used smaller 3D volumetric images of size (64 x 128 x 128) (z * y * x). The training accuracy reached high of 96.55%. In our full prediction script, we use pre-trained models to create predicted labels on full 3D volumetric images of any size. Our prediction script breaks full volumes into smaller sub- volumes to run a prediction on, and then stitches the predicted sub-volume labels together to form the full segmented label. We ensured that there are no edge artifacts by developing a customizable overlap feature. For our full predictions, we attained best results with an overlap (OL) of 10 pixels. **(B)** ECDF for the predicted segmentation label which displays a sizable portion of pixels representing high confidence predictions. **(C)** Heat-map of prediction probability versus ground truth 3D euclidean distance transform. Displays that a majority of high probability points are near to non-zero values in the ground truth. **(D)** Plots for binarizing threshold value versus various segmentation metrics such as IoU (Jaccard Index), Accuracy, Precision, and Recall. **(E)** Plot for training/validation accuracy and loss over 375 epochs. The training was completed in 12.23 hours and the prediction script was ran on 11 full volumes in approximately 56 minutes and 37 seconds. The training used three NVIDIA Titan V GPUs while the prediction script only used one. This process is much more rapid compared to the previous ACME segmentation software which took up to 3.67 hours for 11 full volumes.

Full Predicted Label

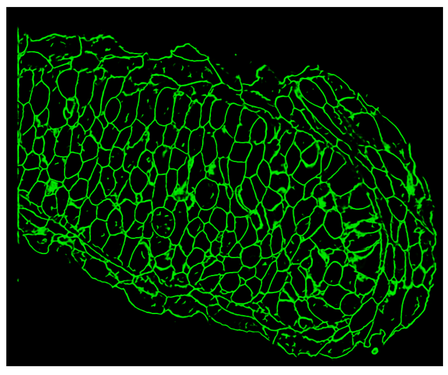

Figure 4. Full Predicted Slice Derived From Raw Data in Figure 2. (A) Displays product of full prediction script on a single slice. Slice corresponds to the raw data displayed in **Figure 2 (C)**. Developing the full segmentation took approximately 5 minutes and 56 seconds using a NVIDIA Titan V GPU. Due to the ground truth not being 100% accurate, the U-Net prediction reaches maximum prediction accuracies of ~96.55%.

References: (1) Özgün Çiçek et al. (2016). 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. arXiv:1606.06650. (2) Gao et al. (2019). Cortical column and whole-brain imaging with molecular contrast and nanoscale resolution. *Science* Vol. 363, Issue 6424. (3) Liu et al. (2018). Observing the cell in its native state: Imaging subcellular dynamics in multicellular organisms. *Science* Vol. 360, Issue 6386. (4) Aguet et al. (2016) Membrane dynamics of dividing cells imaged by lattice light-sheet microscopy. *Molecular Biology of the Cell* 2016 27:22, 3418-3435.