NISNet3D: Three-Dimensional Nuclear Synthesis and Instance Segmentation for Fluorescence Microscopy Images

Liming Wu[1], Alain Chen[1], Paul Salama[2], Kenneth Dunn[3] and Edward Delp[1]

[1]Video and Image Processing Laboratory, School of Electrical and Computer Engineering, Purdue University, West Lafayette, 47907, Indiana, United States.
[2]Department of Electrical and Computer Engineering, Indiana University-Purdue University Indianapolis, Indianapolis, 46202, Indiana, United States.
[3]School of Medicine, Indiana University, Indianapolis, 46202, Indiana, United States.

Contributing authors: wu1114@purdue.edu; chen2967@purdue.edu; psalama@iupui.edu; kwdunn@iu.edu; ace@ecn.purdue.edu;

Abstract

The primary step in tissue cytometry is the automated distinction of individual cells (segmentation). Since cell borders are seldom labeled, researchers generally segment cells by their nuclei. While effective tools have been developed for segmenting nuclei in two dimensions, segmentation of nuclei in three-dimensional images remains a challenging task for which few tools have been developed. The lack of effective methods for three-dimensional segmentation represents a bottleneck in the realization of the potential of tissue cytometry, particularly as methods of tissue clearing present researchers with the opportunity to characterize entire organs. Methods based upon deep learning have shown enormous promise, but their implementation is

hampered by the need for large amounts manually-annotated training data. Here we describe 3D Nuclei Instance Segmentation Network (NISNet3D), a deep learning-based approach in which training is accomplished using synthetic data, profoundly reducing the effort required for network training. We compare results obtained from NISNet3D with results obtained from eight existing techniques.

Keywords: Fluorescence microscopy images, generative adversarial networks, nuclear instance segmentation, synthetic microscopy image generation

1 Introduction

Over the past ten years, various technological developments have provided biologists with the ability to collect microscopy images of enormous scale and complexity. Methods of tissue clearing combined with automated confocal or lightsheet microscopes have enabled three-dimensional imaging of entire organs or even entire organisms at subcellular resolution. Novel methods of multiplexing have been developed so that researchers can now simultaneously characterize 50 or more targets in the same tissue. However, as biologists turn to the task of analyzing these extraordinary volumes (tissue cytometry), they quickly discover that the methods of automated image analysis necessary for extracting quantitative data from images of this scale are frequently inadequate to the task. In particular, while effective methods for distinguishing (segmenting) individual cells are available for analyses of two-dimensional images, corresponding methods for segmenting cells in three-dimensional volumes are generally lacking. The problem of three-dimensional image segmentation thus represents a bottleneck in the full realization of tissue cytometry as a tool in biological microscopy.

There are two main categories of segmentation approaches, traditional image processing and computer vision techniques and techniques based on machine learning and in particular deep learning [1, 2]. Traditional techniques (e.g. watershed, thresholding, edge detection, and morphological operations) can be effective, but generally require careful optimization of processing parameters so that settings are seldom robust, even across images to be pooled or compared. Segmentation techniques based on deep learning have shown great promise, in some cases providing accurate and robust results across a range of image types [3–7]. However, their utility is limited by the large amounts of manually annotated (ground truth) data needed for training, validation, and testing. Annotation is a labor-intensive and time-consuming process, especially for a 3D volume. While tools have been developed to facilitate the laborious process of manual annotation [8–11], the generation of training data remains a major obstacle to implementing segmentation approaches based upon deep learning.

To some degree, the problem of generating sufficient training data can be alleviated using data augmentation, a process in which existing manually-annotated training data is supplemented with synthetic data generated from modifications of the manually annotated data [12–15]. An alternative approach is to use synthetic data for training [14, 16, 17]. In [18], 2D distributions of fluorescent markers are generated using GANs, and 3D microscopy volumes are generated by stacking the 2D synthetic image slices. Similarly, in [19], a 3D GAN was used to generate fully 3D volumetric cell masks with variability matching real volumes. We have shown that Generative Adversarial Networks (GANs) can be used to generate synthetic microscopy volumes that can be used for training [7, 20–22], and incorporated this approach into the DeepSynth segmentation system [6].

Here we describe the 3D Nuclei Instance Segmentation Network (NISNet3D), a deep learning-based segmentation technique that use synthetic volumes, manually annotated volumes or a combination of synthetic and annotated volumes. NISNet3D is a true 3D segmentation method based on a 3D Convolutional Neural Network (CNN). CNNs have had great success for solving problems such as object classification, detection, and segmentation [23, 24] and the encoder-decoder architectures have been widely used for biomedical image analysis including volumetric segmentation [25–27] and medical image registration [28]. CNNs have also been developed for nuclear segmentation [3, 6, 15, 29–33] but are designed for segmentation of two-dimensional images and either cannot be used for segmentation of 3D volumes [29, 32, 34] or involve a process in which objects segmented in two dimensional images are fused together to form 3D objects [3, 14, 30] that fail to represent the 3D anisotropy of microscope images. In contrast, NISNet3D is a true three-dimensional segmentation system that operates directly on 3D volumes, using 3D CNNs to exploit 3D information in a microscopy volume, thereby generating more accurate segmentations of nuclei in 3D image volumes.

We demonstrate that NISNet3D can accurately segment individual nuclei using five different types of microscopy volumes. The qualitative and quantitative evaluation results show that NISNet3D achieves promising results on nuclei instance segmentation with and without the use of manual ground truth annotations when compared to other approaches.

We summarize the contributions of this paper as follows:

- We designed a fully convolutional neural network with residual concatenation and self-attention mechanism for nuclei instance segmentation for 3D volumes. We proposed new 3D markers for nuclei instance segmentation using 3D marker-controlled watershed.
- NISNet3D can use annotated volumes and/or synthetic microscopy volumes for training and can analyze large 3D microscopy volume using a divide-and-conquer inference strategy.
- We present three error/difference visualization methods for visualizing segmentation errors in large 3D microscopy volumes without the need of ground truth annotations.

- We conducted experiments on a variety of microscopy volumes using multiple evaluation metrics, and compared NISNet3D to other deep learning image segmentation methods to demonstrate the effectiveness of our approach.

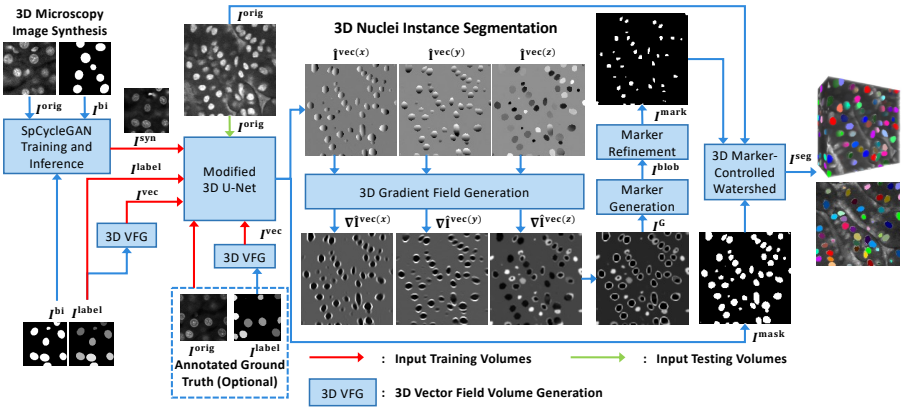

Fig. 1 Overview of NISNet3D for 3D nuclei instance segmentation. Note: The training and synthetic image generation are also shown but are not explicitly part of NISNet3D. NISNet3D is trained with synthetic microscopy volumes generated from SpCycleGAN and/or with annotated actual volumes as indicated

2 PROPOSED METHOD

The block diagram of our proposed nuclei instance segmentation system is shown in Figure 1, which includes: (1) 3D microscopy image synthesis and annotated data, (2) NISNet3D training and inference, and (3) 3D nuclei instance segmentation.

2.1 Notation and Overview

In this paper, we denote I as a 3D image volume of size $X \times Y \times Z$ voxels, and $I_{(x,y,z)}$ is a voxel having coordinate (x, y, z) in volume I. We will use superscripts to indicate the types of image volumes. For example, I^{orig}, I^{bi} and I^{syn} denote the original microscopy volumes, binary segmentation masks and synthetic microscopy volumes, respectively. I^{label} is the gray-scale label volume for I^{bi} where different nuclei are marked with unique pixel intensities. This is done to distinguish each nuclei instance. In addition, I^{target} is a four-channel volume of size $4 \times X \times Y \times Z$, where the first three channels denote the 3D vector field volume I^{vec} that contains the nuclei shape and size information, and the last channel is the 3D binary masks I^{bi}. To represent the 3D nuclei centroid and boundary information in I^{vec}, each nuclei voxel $I^{\text{vec}}_{(x,y,z)}$ denotes a 3D vector $\vec{\mathbf{V}}_{(x,y,z)}$ that points to its nearest nucleus centroid. The details of 3D

vector field generation will be discussed in Section 2.3.1. I^{mask} and $\hat{I}^{\text{vec}}$ are the output of NISNet3D where I^{mask} is the 3D binary segmentation mask and $\hat{I}^{\text{vec}}$ is the estimated 3D vector field volume. Note that $\hat{I}^{\text{vec}(x)}$, $\hat{I}^{\text{vec}(y)}$, and $\hat{I}^{\text{vec}(z)}$ represent the x, y, and z channel of $\hat{I}^{\text{vec}}$, respectively. $\hat{I}^{\text{vec}}$ is then decoded to generate the gradient volume I^{G} where nuclei boundaries are highlighted. I^{mark} is the high quality markers generated from I^{G}. To separate touching nuclei, we use 3D marker-controlled watershed segmentation with the pairs of I^{mark} and I^{mask}, which will be discussed in Section 2.3.2. The post-processing includes small object removal and nuclei color coding for visualization. I^{seg} is the final color-coded segmentation volume. The overview of our proposed approach is shown in Figure 1.

2.2 3D Microscopy Image Synthesis

Deep learning methods generally require large amounts of training samples to achieve accurate results. However, manually annotating ground truth is a tedious task and impractical in many situations especially for 3D microscopy volumes. To address this issue, NISNet3D can use synthetic microscopy 3D volumes for training the segmentation network. It must be emphasized that NISNet3D can use synthetic volumes or annotated real volumes or combinations. We demonstrate this in the experiments.

For generating synthetic volumes we first generate synthetic segmentation masks which are used as the ground truth masks, and further translate the synthetic segmentation masks to synthetic microscopy volumes using an unsupervised image-to-image translation model known as SpCycleGAN [7].

2.2.1 Synthetic Segmentation Mask Generation

We first generate synthetic binary nuclei segmentation mask I^{bi} by iteratively adding candidate binary nucleus I^{nuc} to an empty 3D volume of size $128 \times 128 \times 128$. To synthesize ellipsoidal nuclei, the candidate nuclei are modeled as 3D binary ellipsoids with random size and orientation parameterized by $\boldsymbol{a}$ and $\boldsymbol{\theta}$. Then we iteratively generate N candidate nuclei in different size and orientation, where a range these parameters are randomly selected based on the observation of nuclei characteristics in actual microscopy volume (e.g. nuclei size, shape, and orientation).

The nuclei size is parameterized by the semi-axes length $\boldsymbol{a} = [a_x, a_y, a_z]^T$ of an ellipsoid, the orientation is defined by a rotation angle $\boldsymbol{\theta} = [\theta_x, \theta_y, \theta_z]^T$, and the location is represented by a translation vector $\boldsymbol{t} = [t_x, t_y, t_z]^T$ towards the origin. Equation 1 defines the k^{th} candidate nucleus with voxel intensity $k \in \{1, ..., N\}$. The voxels are assigned intensity values to differentiate them from each other.

$$I^{\text{nuc}}_{(x,y,z)} = \begin{cases} k, & \text{if } (\frac{x}{a_x})^2 + (\frac{y}{a_y})^2 + (\frac{z}{a_z})^2 < 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Equation 2 defines the translated nucleus coordinates $\tilde{\mathbf{X}}$.

$$\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x) \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \end{bmatrix} \tag{2}$$

where $R_x(\theta_x)$, $R_y(\theta_y)$ and $R_z(\theta_z)$ are rotation matrices around the x, y, z axes with angles θ_x, θ_y, and θ_z respectively. Note that the maximum overlap between two candidate nuclei cannot be more than t_o voxels.

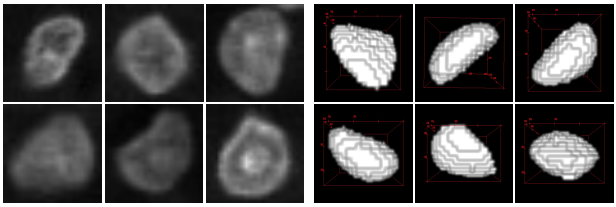

Fig. 2 Non-ellipsoidal shaped nuclei in actual microscopy volumes (left) and synthetic binary nuclei segmentation masks after using elastic deformation (right)

Many of the nuclei are not strictly ellipsoidal. Instead, they look more like deformed ellipsoids (See Figure 2 (left)). To model these non-ellipsoidal nuclei, we use elastic transformation [35] to deform the 3D binary mask of the nuclei. Suppose the 3D binary volume to be deformed is denoted as I^{bi} and is of size $X \times Y \times Z$. We define the amount of deformation using what we will call a displacement vector field. We define the "smooth displacement vector field" I^{smooth} as a matrix of size $3 \times X \times Y \times Z$ to represent three 3D volumes $I^{\text{smooth}(x)}$, $I^{\text{smooth}(y)}$, $I^{\text{smooth}(z)}$ each of size $X \times Y \times Z$. In $I^{\text{smooth}(x)}$, each voxel $I^{\text{smooth}(x)}_{(x,y,z)}$ indicates the distance the voxel $I^{\text{bi}}_{(x,y,z)}$ needs to be shifted on the x-axis. Similarly, for $I^{\text{smooth}(y)}$ and $I^{\text{smooth}(z)}$, each voxel $I^{\text{smooth}(y)}_{(x,y,z)}$ and $I^{\text{smooth}(z)}_{(x,y,z)}$ indicates the distance of the voxel $I^{\text{bi}}_{(x,y,z)}$ that needs to be shifted on the y and z-axis, respectively. We then describe how to construct I^{smooth}.

Next we define the "coarse displacement vector field" I^{coarse} as a matrix of size $3 \times d \times d \times d$, where d controls the amount deformation in the nuclei. Each entry in I^{coarse} is a random variable that is independent and normally distributed $\mathcal{N}(0, \sigma^2)$. The "smooth displacement vector field" I^{smooth} is obtained from I^{coarse} using spline interpolation [36] or bilinear interpolation [35]. The deformation control, d, is used to define the size of I^{coarse}. A larger d will result in more deformation for I^{smooth} whereas lower d indicates less deformation for I^{smooth}. In our experiments, we used spline interpolation and $d \in \{4, 5, 10\}$. Examples of deformed ellipsoids are shown in Figure 2 (right).

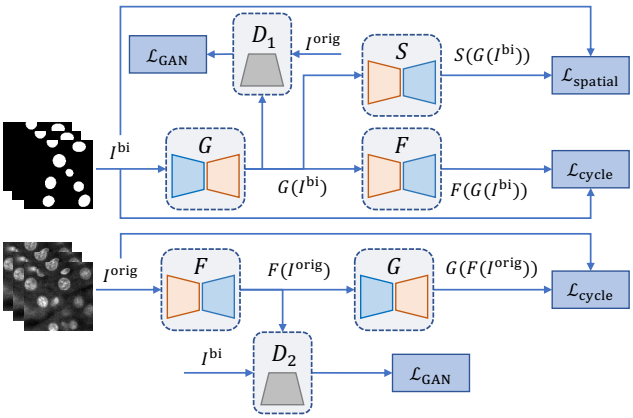

Fig. 3 The architecture of SpCycleGAN that is used for generating synthetic microscopy volumes

2.2.2 Synthetic Microscopy Volume Generation

We use the unpaired image-to-image translation model known as SpCycleGAN [7, 37] for generating synthetic microscopy volumes. By unpaired we mean that the binary segmentation mask we created above is not the ground truth of actual microscopy images. The input to the SpCycleGAN is the binary segmentation masks we created and actual microscopy images (i.e. unpaired). As shown in Figure 1, we use the binary segmentation mask we created I^{bi} and actual microscopy volumes I^{orig} for training the SpCycleGAN. After training we generate synthetic microscopy volumes (I^{syn}) by using different synthetic microscopy segmentation masks (I^{bi}) we created as input to the SpCycleGAN. Note that since the SpCycleGAN generates 2D slices, we then use the slices to construct a 3D synthetic volume. The SpCycleGAN [7], an extension of CycleGAN [37], is shown in Figure 3. SpCycleGAN consists of two generators G and F, two discriminators D_1 and D_2. G learns the mapping from I^{orig} to I^{bi} whereas F performs the reverse mapping. Also, SpCycleGAN introduced a segmentor S for maintaining the spatial location between I^{bi} and $F(G(I^{\text{bi}}))$. The entire loss function of SpCycleGAN is shown in Equation 3.

$$\begin{aligned}\mathcal{L}(G, F, S, D_1, D_2) = &\mathcal{L}_{\text{GAN}}(G, D_1, I^{\text{bi}}, I^{\text{orig}}) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_2, I^{\text{orig}}, I^{\text{bi}}) \\ &+ \lambda_1 \mathcal{L}_{\text{cycle}}(G, F, I^{\text{orig}}, I^{\text{bi}}) \\ &+ \lambda_2 \mathcal{L}_{\text{spatial}}(G, S, I^{\text{orig}}, I^{\text{bi}})\end{aligned} \tag{3}$$

where λ_1 and λ_2 are weight coefficients controlling the loss balance between $\mathcal{L}_{\text{cycle}}$ and $\mathcal{L}_{\text{spatial}}$, and

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_1, I^{\text{bi}}, I^{\text{orig}}) = &\mathbb{E}_{I^{\text{orig}}}[\log(D_1(I^{\text{orig}}))] \\ &+ \mathbb{E}_{I^{\text{bi}}}[\log(1 - D_1(G(I^{\text{bi}})))]\end{aligned}$$

$$\begin{aligned}
\mathcal{L}_{\text{GAN}}(F, D_2, I^{\text{orig}}, I^{\text{bi}}) &= \mathbb{E}_{I^{\text{bi}}}[\log(D_2(I^{\text{bi}}))] \\
&\quad + \mathbb{E}_{I^{\text{orig}}}[\log(1 - D_2(F(I^{\text{orig}})))] \\
\mathcal{L}_{\text{cycle}}(G, F, I^{\text{orig}}, I^{\text{bi}}) &= \mathbb{E}_{I^{\text{bi}}}[\|F(G(I^{\text{bi}})) - I^{\text{bi}}\|_1] \\
&\quad + \mathbb{E}_{I^{\text{orig}}}[\|G(F(I^{\text{orig}})) - I^{\text{orig}}\|_1] \\
\mathcal{L}_{\text{spatial}}(G, S, I^{\text{bi}}, I^{\text{orig}}) &= \mathbb{E}_{I^{\text{bi}}}[\|S(G(I^{\text{bi}})) - I^{\text{bi}}\|_2] \qquad (4)
\end{aligned}$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ denotes the L_1 norm and L_2 norm. $\mathbb{E}_I$ is the expected value over all input volumes of a batch to the network.

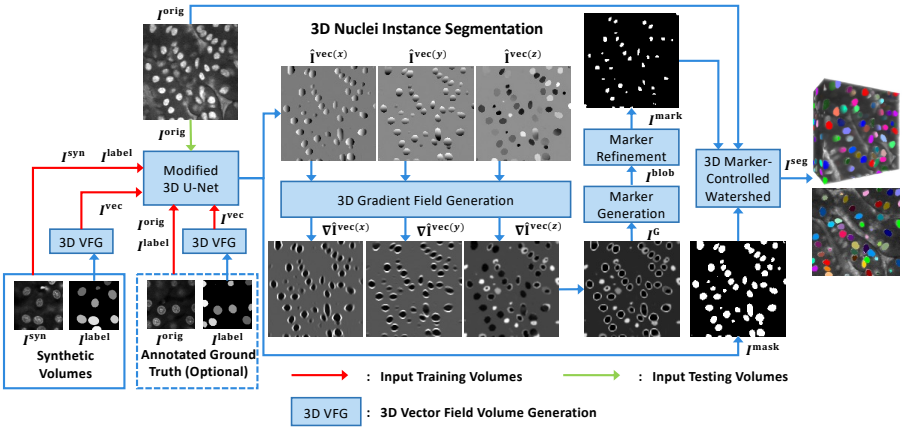

Fig. 4 Overview of NISNet3D - 3D nuclei instance segmentation

2.3 NISNet3D

The overview of NISNet3D is shown in Figure 4. In this section, we describe the architecture of modified 3D U-Net in NISNet3D, how to train and inference the modified 3D U-Net, and nuclei instance segmentation of NISNet3D.

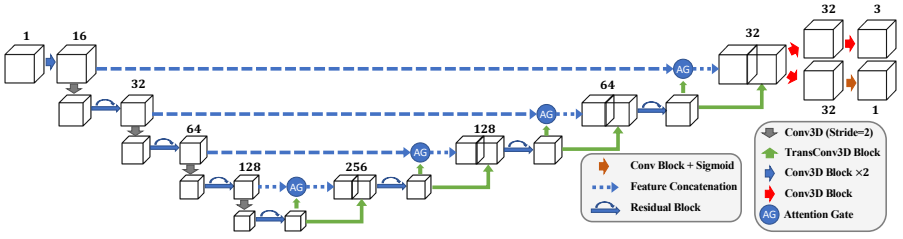

Fig. 5 The proposed NISNet3D uses a modified 3D U-Net architecture with residual blocks, attention gates and multi-task learning module

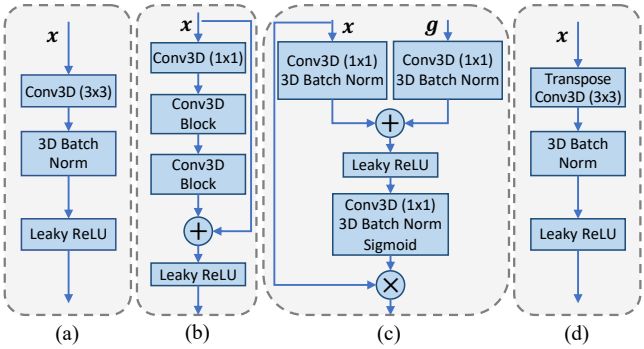

Fig. 6 (a) Conv3D Block, (b) Residual Block, (c) Attention Gate, (d) TransConv3D Block

2.3.1 Modified 3D U-Net

In this section we describe the modified 3D U-Net of NISNet3D as shown in Figure 4. NISNet3D uses an encoder-decoder network (See Figure 5) which outputs the same size volume as the input. The encoder consists of multiple Conv3D Blocks (Figure 6(a)) and Residual Blocks (Figure 6(b)) [38]. Instead of using max pooling layers, we use Conv3D Blocks with stride 2 for feature down-sampling, which introduces more learnable parameters. Each convolution block consists of a 3D convolution layer with filter size $3 \times 3 \times 3$, a 3D batch normalization layer and a leaky ReLU layer. The decoder consists of multiple TransConv3D blocks (Figure 6(d)) and attention gates (Figure 6(c)). Each TransConv3D block includes a 3D transpose convolution with filter size $3 \times 3 \times 3$ followed by 3D batch normalization and leaky ReLU. We use a self-attention mechanism described in [39] to refine the feature concatenation while reconstructing the spatial information.

Training The Modified 3D U-Net. The modified 3D U-Net can be trained on both synthetic microscopy volumes I^{syn} or actual microscopy volumes I^{orig} if manual ground truth annotations are available.

We define the "3D vector field volume," I^{vec}, as a volume where each nucleus voxel is a 3D vector that points to the centroid of the current nucleus in I^{label} that is being used for training. I^{vec} is generated from I^{label} and used as part of the ground truth for training the modified 3D U-Net. We next describe the steps for 3D vector field volume generation (VFG).

As shown in Figure 4, during training, the modified 3D U-Net in NISNet3D takes I^{syn} or I^{orig}, I^{label} and I^{vec} as input, and outputs the estimated 3D vector field volume $\hat{I}^{\text{vec}}$ ($\hat{I}^{\text{vec}(x)}$, $\hat{I}^{\text{vec}(y)}$, $\hat{I}^{\text{vec}(z)}$) and the 3D binary segmentation masks I^{mask}. I^{label} is the gray-scale label volume for I^{bi} with size $X \times Y \times Z$ where different nuclei are marked with unique pixel intensities. As shown in Figure 7, the first step for 3D vector field volume generation (VFG) is to obtain the centroid of each nucleus in I^{label}. We denote the k^{th} nucleus as the voxels with intensity k in I^{label}, and the centroid of k^{th} nucleus is (x_k, y_k, z_k).

I^{vec} is a matrix of size $3 \times X \times Y \times Z$ that represents the three volumes $I^{\text{vec}(x)}$, $I^{\text{vec}(y)}$, $I^{\text{vec}(z)}$ each of size $X \times Y \times Z$. Note that $I^{\text{vec}}_{(x,y,z)} =$

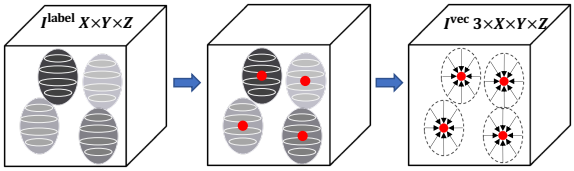

Fig. 7 Steps for 3D vector field volume generation (VFG). Each nucleus voxel in the 3D vector field volume, I^{vec} represents a 3D vector that points to the centroid of current nucleus

$(I^{vec(x)}_{(x,y,z)}, I^{vec(y)}_{(x,y,z)}, I^{vec(z)}_{(x,y,z)})$. We use $I^{vec}_{(x,y,z)}$ to represent a 3D vector at the current location and points to its nearest nucleus centroid. Note that if a voxel $I^{label}_{(x,y,z)}$ is a background voxel (i.e. $I^{label}_{(x,y,z)}$=0), the corresponding voxels $I^{vec}_{(x,y,z)}$ in the vector field volume are set to $\mathbf{0}$. Equation 5 shows the definition of I^{vec}.

$$I^{vec}_{(x,y,z)} = \begin{cases} \vec{\mathbf{V}}^{k}_{(x,y,z)}, & \text{if } I^{label}_{(x,y,z)} \neq 0 \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{5}$$

$\vec{\mathbf{V}}^{k}_{(x,y,z)}$ is a 3D vector from (x, y, z) to (x_k, y_k, z_k). By using the 3D vector field volume, the boundary information is more easily learned since the vectors in boundary regions points to very different directions which will result in very large gradients. I^{vec} is used as part of the training groundtruth for the network to learn nuclei centroid and boundary information whereas I^{label} is used for learning the segmentation masks. Finally, I^{vec} and I^{label} are used as the ground truth for training modified 3D U-Net.

Loss Functions. The modified 3D U-Net simultaneously learns the nuclei segmentation masks I^{mask} and the 3D vector field volume $\hat{I}^{vec}$. Two branches are used and there is no sigmoid function used to obtain $\hat{I}^{vec}$ because the vector represented at a voxel can point to anywhere in a volume. In other words, the voxel in $\hat{I}^{vec}$ can be a negative number or a large number. Unlike previous methods [31, 40, 41] that directly learn the distance transform map, the 3D vector field volume contains both nuclei centroid and boundary information which can avoid over-detection for irregular nuclei. The output 3D vector field volumes $\hat{I}^{vec}$ are compared with the ground truth vector field volume I^{vec} and optimized using the Mean Square Error (MSE) loss function, whereas the segmentation results I^{mask} are compared with the ground truth binary volumes I^{bi} and optimized using the combination of Focal Loss [42] $\mathcal{L}_{FL}$ and Tversky Loss [43] $\mathcal{L}_{TL}$. The entire loss function is shown in Equation 6,

$$\begin{aligned} \mathcal{L}(S, \hat{S}, V, \hat{V}) = \lambda_3 \mathcal{L}_{TL}(S, \hat{S}) + \lambda_4 \mathcal{L}_{FL}(S, \hat{S}) \\ + \lambda_5 \mathcal{L}_{MSE}(V, \hat{V}) \end{aligned} \tag{6}$$

where

$$\mathcal{L}_{\text{TL}} = \frac{\sum_{i=1}^{P} s_{i1}\hat{s}_{i1}}{\sum_{i=1}^{P} s_{i1}\hat{s}_{i1} + \alpha_1 \sum_{i=1}^{P} s_{i1}\hat{s}_{i0} + \alpha_2 \sum_{i=1}^{P} s_{i0}\hat{s}_{i1}},$$
$$\mathcal{L}_{\text{FL}} = -\frac{1}{P}\sum_{i=1}^{P} \{\beta s_{i1}\hat{s}_{i0}^{\gamma}\log(\hat{s}_{i1}) + (1-\beta)s_{i0}\hat{s}_{i1}^{\gamma}\log(\hat{s}_{i0})\},$$
$$\mathcal{L}_{\text{MSE}} = \frac{1}{P}\sum_{i=1}^{P}(v_i - \hat{v}_i)^2 \tag{7}$$

where $\alpha_1 + \alpha_2 = 1$ are two hyper-parameters in Tversky loss [43] that controls the balance between false positive and false negative detections. β and γ are two hyper-parameters in Focal loss [42] where β balances the importance of positive/negative voxels, and γ adjusts the weights for easily classified voxels. In addition, S is the ground truth binary volume, $\hat{S}$ is the segmentation volume, V is the ground truth vector field volume and $\hat{V}$ is the estimated vector field volume. $v_i \in V$ is the i^{th} voxel in V, and $\hat{v}_i \in \hat{V}$ is the i^{th} voxel in $\hat{V}$. Similarly, $s_i \in S$ is the i^{th} voxel in S, and $\hat{s}_i \in \hat{S}$ is the i^{th} voxel in $\hat{S}$. We define $\hat{s}_{i0}$ as the probability that the i^{th} voxel in $\hat{S}$ is the nuclei class, and $\hat{s}_{i1}$ as the probability that the i^{th} voxel in $\hat{S}$ is the background class. Similarly, $s_{i1} = 1$ if s_i is a nuclei voxel and 0 if s_i is a background voxel, and vice versa for s_{i0}. Lastly, P is the total number of voxels in a volume.

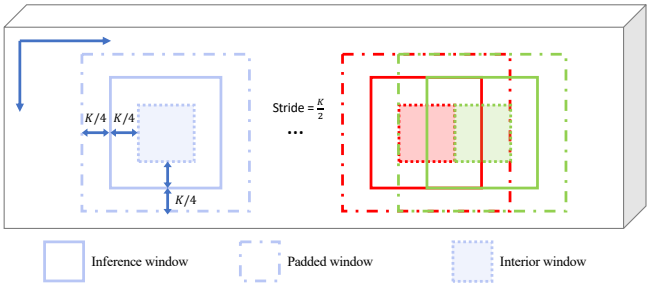

Fig. 8 Proposed divide-and-conquer inference scheme for segmenting large microscopy volumes

Modified 3D U-Net Inference. To segment a large microscopy volume, we propose an divide-and-conquer inference scheme shown in Figure 8. We use an inference window of size $K \times K \times K$ that slides along the original microscopy volume I^{orig} of size $X \times Y \times Z$ and crops a subvolume. Considering that the partially included nuclei on the border of the inference window may cause inaccurate segmentation results, we construct a padded window by symmetrically padding each cropped subvolume by $\frac{K}{4}$ voxels on each border. Also, the stride of the moving window is set to $\frac{K}{2}$ so every step it slides, it will have a $\frac{K}{2}$ voxel overlapping with the previous window. For the inference

results of every $K \times K \times K$ window, only the interior $\frac{K}{2} \times \frac{K}{2} \times \frac{K}{2}$ subvolume, denoted as interior window, in the center will be used as the segmentation results. In this paper, we use $K = 128$ for all testing data. Once the inference window slides along the entire volume, a segmentation volume I^{mask} and 3D vector field volume $\hat{I}^{\text{vec}}$ of size $X \times Y \times Z$ will be generated. In this way, we can inference on any size input volume, especially very large volumes. I^{mask} is the binary segmentation results whereas $\hat{I}^{\text{vec}}$ is the estimated 3D vector field volume that needs to be decoded to locate the nuclei centroids using 3D nuclei instance segmentation.

2.3.2 3D Nuclei Instance Segmentation

Figure 4 shows an overview of the proposed method for 3D nuclei instance segmentation. Based on the output of the modified 3D U-Net, a 3D gradient field generation, marker generation and marker refinement step is used for separating densely clustered nuclei. The estimated 3D vector field volume $\hat{I}^{\text{vec}}$ is a 3-channel volume with the same size as the ground truth 3D vector field volume I^{vec} where each nuclei voxel represents a 3D vector pointing to its nearest nucleus centroid. The neighbor voxels on the boundary of two touching nuclei generally point to different directions and thus have a large gradient. Let $\nabla\hat{I}^{\text{vec}}$ be the gradient of $\hat{I}^{\text{vec}}$. We use $\hat{I}^{\text{vec}}$ to obtain the gradient map I^{G} as described in Equation 8,

$$\begin{aligned}\nabla\hat{I}^{\text{vec}} &= \left[\nabla\hat{I}^{\text{vec}(x)} \;\; \nabla\hat{I}^{\text{vec}(y)} \;\; \nabla\hat{I}^{\text{vec}(z)}\right]^T \\ &= \left[\frac{\partial\hat{I}^{\text{vec}(x)}}{\partial x} \;\; \frac{\partial\hat{I}^{\text{vec}(y)}}{\partial y} \;\; \frac{\partial\hat{I}^{\text{vec}(z)}}{\partial z}\right]^T \\ &= \left[S_x * \hat{I}^{\text{vec}(x)} \;\; S_y * \hat{I}^{\text{vec}(y)} \;\; S_z * \hat{I}^{\text{vec}(z)}\right]^T \end{aligned} \tag{8}$$

where $\hat{I}^{\text{vec}(x)}$, $\hat{I}^{\text{vec}(y)}$, and $\hat{I}^{\text{vec}(z)}$ are the x, y, and z channel of the estimated 3D vector field volume $\hat{I}^{\text{vec}}$. S_x, S_y, S_z are 3D Sobel filters and $*$ is the convolution operator. In Equation 9, the gradient map I^{G} is then determined by choosing the maximum gradient component of each vector on the x, y and z direction.

$$I^{\text{G}} = \max(\nabla\hat{I}^{\text{vec}(x)}, \nabla\hat{I}^{\text{vec}(y)}, \nabla\hat{I}^{\text{vec}(z)}) \tag{9}$$

The boundary of touching nuclei in I^{G} have larger values (larger gradients) which can be used to identify individual nucleus. In Equation 10, I^{mask} is the binary segmentation mask from the modified 3D U-Net and $\tau(x, t)$ is a thresholding function such that $\tau(x, t) = 1$ if $x \geq t$ otherwise 0. $\tau(I^{\text{G}}, T_{\text{m}})$ is used to highlight the boundary of individual nuclei and $\sigma(*)$ is a rectifier that sets all negative values to 0. I^{blob} is the interior regions of the nuclei which are potential markers for watershed segmentation [44, 45]

$$I^{\text{blob}} = \sigma(I^{\text{mask}} - \tau(I^{\text{G}}, T_{\text{m}})) \tag{10}$$

$$I^{\text{mark}} = \delta_{t_{\text{f}}}(\delta_{t_{\text{c}}}(I^{\text{blob}}, B_{\text{c}}), B_{\text{f}}) \tag{11}$$

To better refine I^{blob}, we use a 3D conditional erosion with a coarse structuring

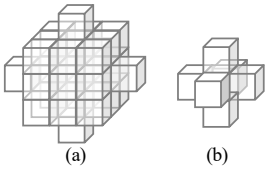

Fig. 9 (a) coarse 3D structuring element B_{c} and (b) fine 3D structuring element B_{f} for 3D conditional erosion

element B_{c} and fine structuring B_{f} shown in Figure 9. We first iteratively erode each object using B_{c} until the object size is smaller than t_{c} then erode each object using B_{f} until the size of each object is smaller than t_{f}. The markers I^{mark} for watershed segmentation are obtained using Equation 11 where $\delta_{t_{\text{c}}}(I^{\text{blob}}, B_{\text{c}})$ defines the iterative erosion of all objects in I^{blob} with coarse structuring element B_{c} until the size of each object is smaller than coarse object threshold t_{c}. Similarly, the output of $\delta_{t_{\text{c}}}(I^{\text{blob}}, B_{\text{c}})$ is then eroded with fine structuring element B_{f} and fine object threshold t_{f}. Finally, marker-controlled watershed [45] is used to generate instance segmentation masks I^{seg}. Small objects less than 20 voxels are then removed, and each object is color coded for visualization.

Table 1 The description of the five datasets used in the evaluations

Original Microscopy Volumes			
Volume Name	**Source**	**Original Size ($X \times Y \times Z$)**	**Annotated Subvolume Size**
$\mathcal{V}_1$	cleared rat kidney	$512 \times 512 \times 200$	$128 \times 128 \times 64$
$\mathcal{V}_2$	rat liver	$512 \times 512 \times 32$	$128 \times 128 \times 32$
$\mathcal{V}_3$	cleared rat kidney	$512 \times 512 \times 415$	$128 \times 128 \times 64$
$\mathcal{V}_4$	cleared mouse intestine	$512 \times 930 \times 157$	$128 \times 128 \times 40$
$\mathcal{V}_5$	zebrafish brain	$2000 \times 1450 \times 397$	$64 \times 64 \times 64$

3 EXPERIMENTAL RESULTS

3.1 Evaluation Datasets

Due to the limited availability of annotated volumes, we present two training and evaluation strategies described in Section 3.2. The corresponding trained versions of NISNet3D obtained using the two strategies are denoted as "NISNet3D-slim" and "NISNet3D-synth". Both of the versions are evaluated on four different microscopy volumes denoted as $\mathcal{V}_1$-$\mathcal{V}_4$, which are fluorescent-labeled (Hoechst 33342 stain) nuclei collected from rat kidneys, rat livers, and mouse intestines using confocal microscopy. The manually annotated ground

truth subvolumes for each type of the evaluation volumes were obtained using ITK-SNAP [8]. In addition, we also trained NISNet3D-slim on a publicly available electron microscopy volume of a zebrafish brain. This volume is known as NucMM[46] and will be denoted as $\mathcal{V}_5$ in our evaluation datasets. The detailed information of all five datasets used in our evaluation is shown in Table 1.

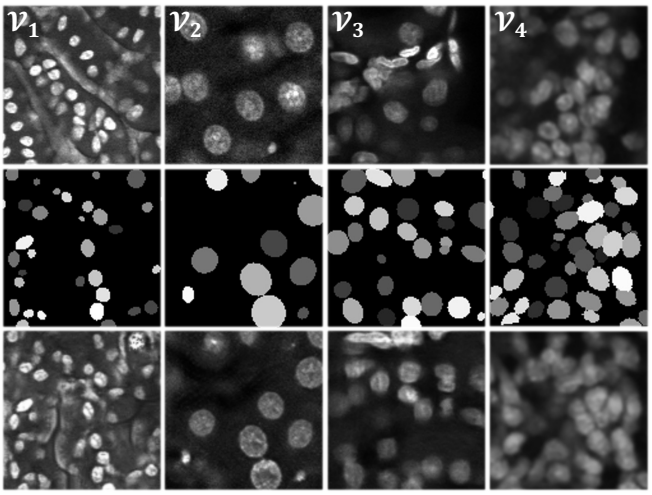

Fig. 10 Synthetic microscopy images (third row) used for training. The synthetic nuclei segmentation masks (second row) are based on the actual microscopy images (first row)

3.2 Experimental Settings

The parameters for generating I^{bi} are shown in Table 2 where ($a_{\min}$, $a_{\max}$) is the range of ellipsoids semi-axes $\mathbf{a}$. t_{o} is the maximum allowed overlapping voxels between two nuclei and N is the total number of nuclei in a synthetic volume. These parameters are based on visual inspection of nuclei characteristics in the actual microscopy volumes. The synthetic microscopy volumes were verified by a biologist (one of the co-authors). The SpCycleGAN was trained on unpaired I^{bi} and I^{orig} and the trained model was used to generate synthetic microscopy volumes. The weight coefficient of the loss function are set to $\lambda_1 = \lambda_2 = 10$ based on the experiments described in [7].

Both the SpCycleGAN and NISNet3D are implemented using PyTorch. We used 9-block ResNet for generators G, F and the segmentor S (see Figure 3). The discriminators D_1 and D_2 (Figure 3) are implemented with the "PatchGAN" classifier [47]. The SpCycleGAN was trained with Adam optimizer [48] for 200 epochs with initial learning rate 0.0002 that linearly decays to 0 after the first 100 epochs. Figure 10 shows the generated synthetic nuclei segmentation masks and corresponding synthetic microscopy images.

For NISNet3D-slim, we trained 5 versions denoted as $\mathcal{M}_1$-$\mathcal{M}_5$ corresponding to $\mathcal{V}_1$-$\mathcal{V}_5$ (See Table 2). We used three training methods for NISNet3D-slim:

Table 2 The training evaluation strategies for NISNet3D-slim and comparison methods

Parameters For Generating Synthetic Microscopy Volumes							**NISNet3D-slim Training Strategies**		**NISNet3D-slim**
Volume Name	$a_{\min}$	$a_{\max}$	t_o	N	d	σ	**Version**	**Training**	**Evaluation**
$\mathcal{V}_1$	4	8	5	500	10	1	$\mathcal{M}_1$	50 volumes of synthetic $\mathcal{V}_1$	Tested on 1 subvolume of $\mathcal{V}_1$
$\mathcal{V}_2$	10	14	10	70	4	1	$\mathcal{M}_2$	50 volumes of synthetic $\mathcal{V}_2$	Tested on 16 subvolumes of $\mathcal{V}_2$
$\mathcal{V}_3$	6	10	200	200	5	2	$\mathcal{M}_3$	50 volumes of synthetic $\mathcal{V}_3$	3-fold cross-validated on 9 subvolumes of $\mathcal{V}_3$
$\mathcal{V}_4$	8	10	100	560	4	4	$\mathcal{M}_4$	lightly retrained using $\mathcal{M}_3$ on 1 subvolume of $\mathcal{V}_4$	Tested on 4 subvolumes of $\mathcal{V}_4$
$\mathcal{V}_5$	-	-	-	-	-	-	$\mathcal{M}_5$	27 subvolumes of $\mathcal{V}_5$	Tested on 27 subvolumes of $\mathcal{V}_5$

(1) train only on corresponding synthetic microscopy data ($\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$), (2) transfer the weights from $\mathcal{M}_3$ of NISNet3D-slim and continue training on a limited number of actual microscopy subvolumes ($\mathcal{M}_4$), (3) directly train on only actual subvolumes (for $\mathcal{M}_5$).

After training, two different evaluation schemes are used for NISNet3D-slim: (1) directly test on all subvolumes of original volume (for $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_4$, $\mathcal{M}_5$), (2) use 3-fold cross-validation: split ground truth subvolumes randomly into 3 equal sets and iteratively update the model on one of the set and test on other two sets (for $\mathcal{M}_3$). We use cross-validation in our experiments to show the effectiveness of our method when the evaluation data is limited. Note that for lightly retraining, we update all parameters of NISNet3D while continue training on actual microscopy volumes. The training and evaluation scheme for each model is shown in Table 2.

In addition, we also trained NISNet3D-synth. This is trained on 800 synthetic microscopy volumes including synthetic $\mathcal{V}_1$-$\mathcal{V}_4$ without updating using any actual microscopy volumes. NISNet3D-synth is designed for the scenario where no ground truth annotations are available. NISNet3D-slim is used for the case where limited ground truth annotated volumes are available. In this case synthetic volumes are used for training and the small amount of ground truth data is for used light retraining.

Table 3 Parameters used for NISNet3D nuclei instance segmentation

	NISNet3D-slim					**NISNet3D-synth**			
Parameters	$\mathcal{V}_1$	$\mathcal{V}_2$	$\mathcal{V}_3$	$\mathcal{V}_4$	$\mathcal{V}_5$	$\mathcal{V}_1$	$\mathcal{V}_2$	$\mathcal{V}_3$	$\mathcal{V}_4$
T_{m}	5	5	1	1	0	0	0	4	0
T_{c}	700	3000	2000	2000	2000	2000	2000	2000	2000
T_{f}	200	500	700	300	200	200	200	500	200

Both NISNet3D-slim and NISNet3D-synth were trained for 100 epochs using the Adam optimizer [48] with constant learning rate 0.001. The weight coefficients of the loss function are set to $\lambda_3 = 1$, $\lambda_4 = 10$, $\lambda_5 = 10$. The hyper-parameters β, γ of $\mathcal{L}_{\mathrm{FL}}$ are set to 0.8 and 2, and the hyper-parameters α_1, α_2 in $\mathcal{L}_{\mathrm{TL}}$ are set to 0.3 and 0.7 [43, 49]. The nuclei instance segmentation parameters used in our experiments are shown in Table 3.

3.3 Comparison Methods

We compared the NISNet3D with both deep learning image segmentation methods including VNet [27], 3D U-Net [26], Cellpose [3], DeepSynth [6], and StarDist3D [15]. In addition, we also compare NISNet3D with several commonly used biomedical image processing tools including 3D Watershed [45], Squassh [50], CellProfiler [51], and VTEA [52].

VNet [27] and 3D U-Net [26] are two popular 3D encoder-decoder networks with shortcut concatenations designed for biomedical image segmentation. Cellpose [3] uses a modified 2D U-Net for estimating image segmentation and

Table 4 Comparison of object-based evaluation metrics for microscopy datasets $\mathcal{V}_1$ - $\mathcal{V}_4$

Methods	**Microscopy $\mathcal{V}_1$**				**Microscopy $\mathcal{V}_2$**				**Microscopy $\mathcal{V}_3$**				**Microscopy $\mathcal{V}_4$**			
	mP	mR	mF_1	mAP	mP	mR	mF_1	mAP	mP	mR	mF_1	mAP	mP	mR	mF_1	mAP
3D Watershed[45]	73.77	63.60	68.31	48.87	70.13	83.80	76.36	60.30	78.43	58.40	66.95	46.73	27.51	33.66	30.22	11.96
Squassh[50]	61.49	14.56	23.54	9.21	84.08	73.15	78.24	61.80	65.48	16.53	26.40	12.26	2.96	11.95	4.73	0.49
CellProfiler[51]	61.67	57.31	59.41	37.79	69.12	82.20	75.09	57.52	59.18	42.46	49.44	26.87	27.77	27.86	27.75	10.15
VTEA[52]	68.91	55.76	61.64	42.71	74.44	76.73	75.57	57.76	43.36	51.56	47.11	23.00	30.23	52.90	38.45	17.89
VNet[27]	91.47	74.98	82.41	70.29	80.52	53.95	64.61	45.88	79.85	70.04	74.56	58.33	36.12	50.26	41.73	23.74
3D U-Net[26]	92.12	73.57	81.81	69.63	86.08	65.85	74.62	58.12	85.40	73.32	78.89	64.21	38.82	51.43	43.93	25.19
Cellpose[3]	89.47	89.47	89.47	81.70	66.07	67.79	66.66	47.80	81.98	78.58	80.24	67.19	58.56	45.01	50.67	30.43
DeepSynth[6]	81.09	94.28	87.19	79.51	71.86	88.75	79.42	66.19	83.29	73.14	77.85	62.86	45.45	47.37	46.34	23.57
NISNet3D-slim	88.69	91.21	**89.93**	**81.97**	81.25	87.95	**84.47**	**72.16**	82.73	83.31	**83.00**	**71.18**	56.40	60.69	**58.43**	**37.30**
Cellpose-synth[3]	74.26	92.31	82.31	72.35	73.64	86.57	79.57	67.59	69.40	72.69	70.98	53.62	53.24	43.53	47.80	26.39
StarDist3D-synth[15]	85.77	74.67	79.83	66.81	63.78	57.54	60.47	40.00	74.73	48.12	58.40	39.52	74.73	48.12	58.40	39.52
NISNet3D-synth	94.36	91.67	**93.00**	**88.56**	71.90	91.85	**80.66**	**68.63**	63.35	81.87	**71.43**	**54.47**	62.69	69.07	**65.73**	**51.60**

Table 5 Comparison of object-based evaluation results for microscopy dataset $\mathcal{V}_5$

Methods	Microscopy $\mathcal{V}_5$						
	mP	mR	mF_1	$AP_{.50}$	$AP_{.75}$	mAP	AJI
StarDist3D[15]	73.44	74.24	73.84	88.59	9.78	61.20	68.56
DeepSynth[6]	81.63	76.04	78.72	71.47	50.88	63.74	75.54
Cellpose[3]	96.15	94.47	95.30	94.90	83.82	91.21	81.31
NISNet3D-slim	**96.89**	**96.24**	**96.56**	**95.98**	**88.84**	**93.62**	**83.90**

spatial flows, and uses a dynamic system to cluster the pixels and further separate touching nuclei. When segmenting 3D volumes, Cellpose works from three different directions slice by slice and reconstruct the 2D segmentation results to a 3D segmentation volume [3]. DeepSynth uses a modified 3D U-Net to segment a 3D microscopy volume and uses watershed to separate touching nuclei [6]. StarDist3D uses a modified 3D U-Net to estimate the star-convex polyhedra to represent the nuclei [15]. 3D Watershed [45] uses the watershed transformation [44] and conditional erosion [45] for nuclei instance segmentation. Squassh is a ImageJ plugin for both 2D and 3D microscopy image segmentation based on the the use active contours [50]. CellProfiler is a image processing toolbox and provides customized image processing and analysis modules [51]. VTEA is an ImageJ plugin that combines Otsu's thresholding and watershed to segment 2D nuclei slice by slice and reconstruct the results to a 3D volume [52].

We trained and evaluated the comparison methods using the same dataset as used for NISNet3D. We also used the same training and evaluation strategies as NISNet3D described in Section 3.2. This is discussed in Section 3.6. Note that 3D Watershed, Squassh, CellProfiler, and VTEA do not need to be trained because they use more traditional image analysis for these techniques which we describe in Section 3.6.

3.4 Evaluation Metrics

We use object-based metrics to evaluate nuclei instance segmentation accuracy. We define N^t_{TP} as the number of True Positive detection where the Intersection-over-Union (IoU) between a detected nucleus and a ground truth nucleus is greater than t voxels. Similarly, N^t_{FP} is the number of False Positives, and N^t_{FN} is the number of False Negatives [53, 54]. N^t_{TP} measures how many nuclei in a volume are correctly detected. The higher N^t_{TP} the more accurate the detection method. N^t_{FP} represents the detected nuclei are not actually nuclei which are false detections. N^t_{FN} represents the number of nuclei that did not detected. A precise detection method should have high N^t_{TP} but low N^t_{FP} and N^t_{FN}.

We then construct metrics based N^t_{TP}, N^t_{FP}, N^t_{FN}. To reduce the bias [55], we use the mean Precision ($\mathrm{mP} = \frac{1}{|T_{\mathrm{IoUs}}|}\sum_{t\in T_{\mathrm{IoUs}}}\frac{N^t_{\mathrm{TP}}}{N^t_{\mathrm{TP}}+N^t_{\mathrm{FP}}}$), mean Recall ($\mathrm{mR} = \frac{1}{|T_{\mathrm{IoUs}}|}\sum_{t\in T_{\mathrm{IoUs}}}\frac{N^t_{\mathrm{TP}}}{N^t_{\mathrm{TP}}+N^t_{\mathrm{FN}}}$) and mean F_1 score ($\mathrm{mF}_1 = \frac{1}{|T_{\mathrm{IoUs}}|}\sum_{t\in T_{\mathrm{IoUs}}}\frac{2N^t_{\mathrm{TP}}}{2N^t_{\mathrm{TP}}+N^t_{\mathrm{FP}}+N^t_{\mathrm{FN}}}$) on using multiple IoU thresholds T_{IoUs}. We set $T_{\mathrm{IoUs}} = \{0.25, 0.3, ..., 0.45\}$ for datasets $\mathcal{V}_1$-$\mathcal{V}_4$, and set $T_{\mathrm{IoUs}} =$

$\{0.5, 0.55, ..., 0.75\}$ for datset $\mathcal{V}_5$. The selection of the T_{IoUs} is described in more detail below.

We observed that the nuclei in datasets $\mathcal{V}_1$-$\mathcal{V}_4$ are more challenging to segment than the nuclei in $\mathcal{V}_5$. If we use the same IoU thresholds for evaluating all the datasets, the evaluation accuracy for $\mathcal{V}_1$-$\mathcal{V}_4$ will be much lower than the evaluation accuracy for $\mathcal{V}_5$. Thus, we chose two different sets of IoU thresholds for $\mathcal{V}_1$-$\mathcal{V}_4$, and $\mathcal{V}_5$, respectively.

We also examined commonly used object detection metrics: Average Precision (AP) [56, 57] by estimating the area under the Precision-Recall Curve [58] using the same thresholds for T_{IoUs} as described in the previous paragraph. For example, $\text{AP}_{.25}$ is the average precision with IoU threshold 0.25. The mean Average Precision (mAP) is obtained as $\text{mAP} = \frac{1}{|T_{\text{IoUs}}|} \sum_{t \in T_{\text{IoUs}}} \text{AP}_t$.

In addition, we use the Aggregated Jaccard Index (AJI) [59] to integrate object and voxel errors. The AJI is defined as:

$$\text{AJI} = \frac{\sum_{i=1}^{N} |G_i \cup S_m^i|}{\sum_{i=1}^{N} |G_i \cap S_m^i| + \sum_{j \in U} |S_j|} \tag{12}$$

where G_i denotes the ith nucleus in ground truth volume with total number of N nuclei. S_m^i is the mth connected component in the segmentation mask which has the largest Jaccard Index with G_i, and U is the segmented nuclei without corresponding ground truth. Note that each segmented nucleus with index m cannot be used more than once.

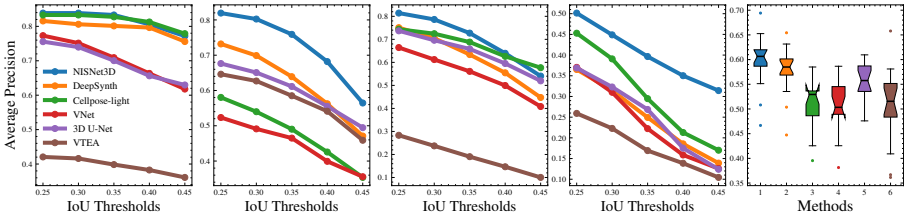

Fig. 11 Evaluation results using Average Precision (AP) for multiple Intersection-over-Unions (IoUs) thresholds, T_{IoUs}, for datasets $\mathcal{V}_1$-$\mathcal{V}_4$, and box plots of Aggregated Jaccard Index (AJI) of each subvolume in dataset $\mathcal{V}_2$

All methods are optimized to achieve the best visual results by parameter tuning. This is further discussed in Section 3.6. The quantitative evaluation metrics for each microscopy datasets are shown in Table 4 and Table 5. Figure 11 shows the AP scores using multiple IoU thresholds and the box plot of AJI on each subvolume of dataset $\mathcal{V}_2$. The orthogonal views (XY focal planes and XZ focal planes) of the segmentation masks that are overlaid on the original microscopy subvolume for each method on $\mathcal{V}_1$-$\mathcal{V}_5$ are shown in Figure 12 and Figure 13. Note the colors correspond to different nuclei.

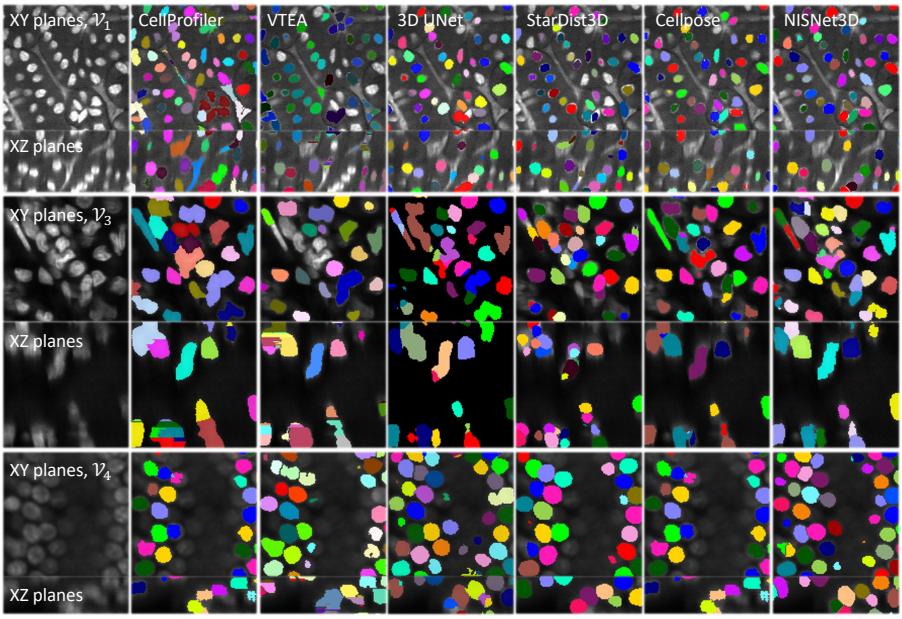

Fig. 12 Visualization of segmentation results for XY and XZ focal planes

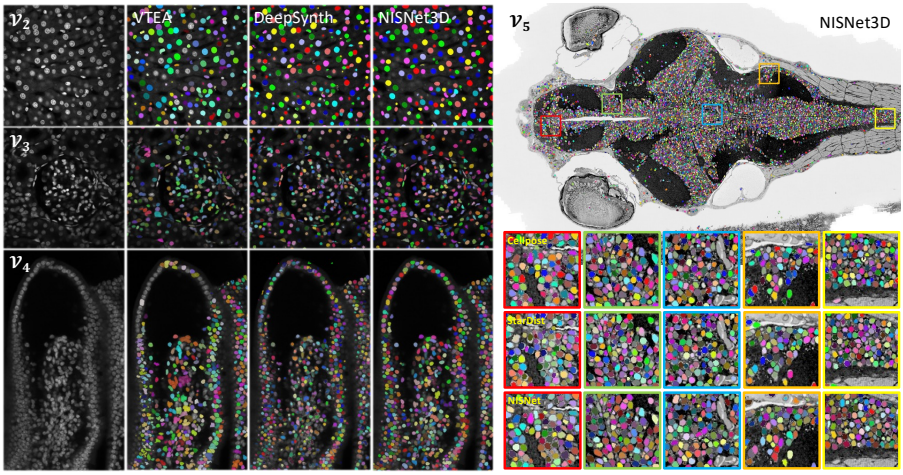

Fig. 13 Visual comparison of the segmentations on the entire volume for datasets $\mathcal{V}_2$, $\mathcal{V}_3$, $\mathcal{V}_4$, and $\mathcal{V}_5$

3.5 Visualizing Errors and Differences

Entire microscopy volumes contain many regions with varying spatial characteristics. In order to see how the segmentation methods perform on various regions, we propose three methods for visualizing the errors and differences between a "test segmented volume" and a "reference segmented volume". We use the NISNet3D segmented volume as the reference segmented volume. We

visualize the segmentation errors between VTEA and NISNet3D and we also visualize the errors between DeepSynth and NISNet3D on entire volumes. We describe how to generate an *Overlay Volume* and three types of *Error Volumes* using the three methods which we will call *Visualization Method A, B*, and *C*. Note that *Visualization Method A* does not need a "reference segmented volume" whereas *Visualization Method B* and *C* need a "reference segmented volume".

Next we describe how to generate an *Overlay Volume*. Using the notation shown in Figure 4, we denote I^{orig} as the original microscopy volume and m as the maximum intensity of I^{orig}. We will use I^{orig} for overlaying the segmentation errors from the "test segmented volume" to construct the visualization. For a "reference segmented volume", we denote I^{mask} as the binary segmentation masks of I^{orig}, and denote I^{seg} as the color-coded segmentation of I^{orig} with RGB channels $I^{\text{seg}}_{\text{R}}$, $I^{\text{seg}}_{\text{G}}$, $I^{\text{seg}}_{\text{B}}$. Similarly, for a "test segmented volume", we denote C^{bi} as the binary segmentation masks of I^{orig}, and denote C as the color-coded segmentation of I^{orig} with RGB channels C_{R}, C_{G}, C_{B}.

We then denote the *Overlay Volume* as L with RGB channels L_{R}, L_{G}, L_{B}. As shown in Equation 13, the *Overlay Volume* for a "test segmented volume" is generated by adding the original microscopy volume to each of the RGB channels of the color-coded segmented volume.

$$L_{\text{R}} = I^{\text{orig}} + C_{\text{R}}, L_{\text{G}} = I^{\text{orig}} + C_{\text{G}}, L_{\text{B}} = I^{\text{orig}} + C_{\text{B}} \tag{13}$$

We then define the notation we used for generating the three types of *Error Volumes*. To represent the segmented nuclei in a "test segmented volume", let $S^{3\text{D}} = \{s^{3\text{D}}_1, s^{3\text{D}}_2, \ldots s^{3\text{D}}_n\}$ be the set of all 3D segmented nuclei in C, where $s^{3\text{D}}_i$ is a volume with same size of C but only contains the ith segmented 3D nucleus, and let $S^{2\text{D}} = \{s^{2\text{D}}_1, s^{2\text{D}}_2, \ldots s^{2\text{D}}_k\}$ be the set of all 2D objects in C from each XY focal planes, where $s^{2\text{D}}_i$ is a volume with same size of C but only contains the ith segmented 2D nucleus.

Similarly, to represent the segmented nuclei in a "reference segmented volume", we denote I^{seg} as the 3D segmentation volume from NISNet3D, which will be used as the "reference segmented volume". Let $O^{3\text{D}} = \{o^{3\text{D}}_1, o^{3\text{D}}_2, \ldots o^{3\text{D}}_s\}$ be the set of all 3D objects in I^{seg}, and let $O^{2\text{D}} = \{o^{2\text{D}}_1, o^{2\text{D}}_2, \ldots o^{2\text{D}}_j\}$ be the set of all 2D objects in I^{seg} from each slice. Next, we describe how to generate the three types of *Error Volume* using the *Visualization Methods A, B*, and *C*.

3.5.1 Visualization Method A

The *Error Volume* generated by *Visualization Method A* shows voxels in the original microscopy volume that are not segmented by either test or reference methods. The input to *Visualization Method A* is the original microscopy volume and a segmented volume ("test segmented volume" or "reference segmented volume"). Using VTEA as an example: the VTEA segmented volume is subtracted from the original microscopy volume. This is shown in Equation

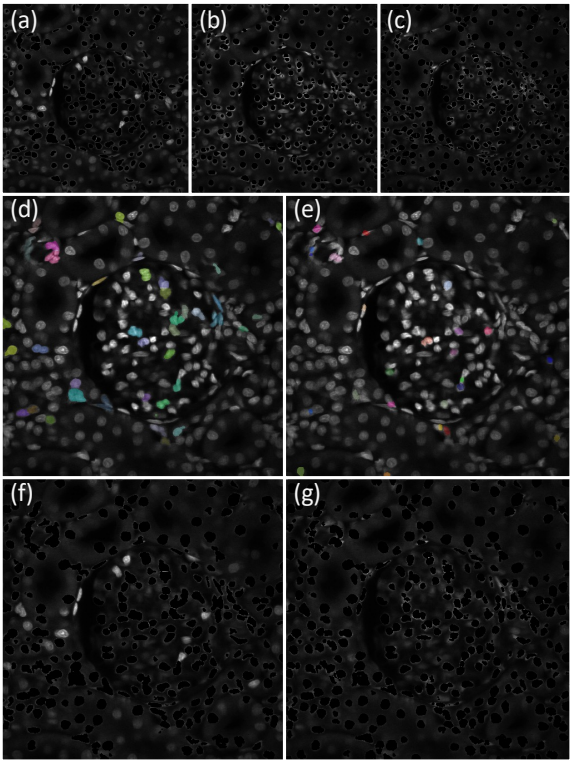

Fig. 14 *Error Volume* for *Visualization Method A* (first row), *Visualization Method B* (second row), and *Visualization Method C* (third row) for VTEA (a)(d)(f), DeepSynth (b)(e)(g), and NISNet3D (c) on microscopy volume $\mathcal{V}_3$

14.

$$I^{\mathrm{A}} = \max(I^{\mathrm{orig}} - m * C^{\mathrm{bi}}, 0) \tag{14}$$

The *Error Volume* I^{A} shows the voxels in the original microscopy image that are not segmented. We can replace C^{bi} in Equation 14 with I^{mask} to obtain the *Error Volume* for the "reference segmented volume" NISNet3D. Figure 14 ((a), (b), (c)) shows the *Error Volumes* generated by *Method A* for VTEA, DeepSynth, and NISNet3D on dataset $\mathcal{V}_3$.

3.5.2 Visualization Method B

The *Error Volume* generated by *Visualization Method B* shows the under-segmentation regions where multiple nuclei in the "reference segmented volume" are detected as a single nucleus in the "test segmented volume". Here we use NISNet3D as "reference segmented volume" and use VTEA or DeepSynth as "test segmented volume". The input to *Visualization Method B* is the VTEA (or DeepSynth) segmented volume and the NISNet3D segmented volume.

Using VTEA as an example: if two or more nuclei in the NISNet3D segmented volume intersect with the same single nucleus in the VTEA segmented volume, then we show the single nucleus segmented by VTEA in the *Visualization Method B Error Volume*. This is shown in the Equation 15.

$$I^{\mathrm{B}} = \bigcup \{ s^{2\mathrm{D}} \in S^{2\mathrm{D}} : \exists o_p^{2\mathrm{D}} \in O^{2\mathrm{D}}, o_q^{2\mathrm{D}} \in O^{2\mathrm{D}}, o_p^{2\mathrm{D}} \neq o_q^{2\mathrm{D}} \\ s^{2\mathrm{D}} \cap o_p^{2\mathrm{D}} \neq \emptyset, s^{2\mathrm{D}} \cap o_q^{2\mathrm{D}} \neq \emptyset \} \tag{15}$$

Then the result volume I^{B} is overlaid on the original microscopy volume using Equation 13. Figure 14 ((d), (e)) shows the *Error Volumes* generated by *Visualization Method B* for VTEA and DeepSynth on dataset $\mathcal{V}_3$.

3.5.3 Visualization Method C

The *Error Volume* generated by *Visualization Method C* shows nuclei segmented by a "reference segmented volume" but are completely missed by a "test segmented volume". The input to *Visualization Method C* is the VTEA (or DeepSynth) segmented volume and the NISNet3D segmented volume. Using VTEA as an example: if the voxels of a nucleus in NISNet3D segmented volume do not intersect with any voxel of any segmented nucleus from the VTEA segmented volume, then the *Visualization Method C Error Volume* will show this nucleus from NISNet3D. This is shown in the Equation 16:

$$I^{\mathrm{C}} = \bigcup \{ o^{3\mathrm{D}} \in O^{2\mathrm{D}} : \neg \exists s^{2\mathrm{D}} \in S^{2\mathrm{D}}, s^{2\mathrm{D}} \cap s^{3\mathrm{D}} \neq \emptyset \} \tag{16}$$

Note: Since we are using the NISNet3D segmented volume as the "reference segmented volume", we do not provide *Error Volume* for *Visualization Methods B* or *C* on the NISNet3D segmented volumes. The *Error Volumes* for *Visualization Method A, B,* and *C* is shown in Figure 14.

3.6 Discussion

Due to the limited availability of annotated volumes, we use synthetic microscopy subvolumes for training NISNet3D. It should emphasized that NISNet3D can be trained on annotated volumes if available or one could use a combination of synthetic volumes and actual annotated volumes. In order to examine the performance of NISNet3D, we tested NISNet3D on four fluorescence microscopy datasets from multiple rat organs and tissue regions. In addition, we also tested electron microscopy data from a zebrafish brain from the NucMM Challenge [46].

We compared NISNet3D with VNet [27], 3D U-Net [26], Cellpose [3], DeepSynth [6], StarDist3D [15], 3D Watershed [45], Squassh [50], CellProfiler [51], and VTEA [52].

For 3D Watershed, we used 3D Gaussian filter to preprocess the image and used Otsu's method to segment the objects from background structure. Then the 3D conditional erosion described in Section 2.3.2 was used to obtain

the markers, and the marker-controlled 3D watershed implemented by Python scikit-image library was used to separate touching nuclei.

For CellProfiler [51], customized image processing modules including inhomogeneity correction, median filtering, and morphological erosion were used to preprocess the image, and default "IdentifyPrimaryObject" module was used to obtain 2D segmentation masks on each slice. Then the 2D segmentations are merged to a 3D segmentation using the blob-slice method described in [52, 60]. For Squassh [50], we used the "Background subtraction" with tuned "rolling ball window size" parameter. The rest of the parameters are set to default. We see that Squassh did fairly well on data $\mathcal{V}_2$ but totally failed on data $\mathcal{V}_4$ due to the densely clustered nuclei. For VTEA [52], Gaussian filter and background subtraction were used to preprocess the image. The object building method is set as "Connect 3D", and segmentation threshold is determined automatically. We tuned the parameters "Centroid offset", "Min vol", and "Max vol" to obtain the best visual segmentation results. Finally, the watershed is chosen for instance segmentation. Since VTEA and Cellprofier's "IdentifyPrimaryObject" module only works on 2D images, we see that their segmentation results shown in Figure 12 suffer from over-segmentation errors on XZ planes.

For VNet [27], 3D U-Net [26] and DeepSynth [6] methods, we improved the segmentation results by using our 3D conditional erosion described in Section 2.3.2 followed by 3D marker-controlled watershed to split the touch nuclei. For Cellpose, we use the "nuclei" style and since the training of Cellpose is only limited on 2D images, we trained Cellpose on every XY focal planes of our subvolumes follow the training schemes in Table 2. We observe that

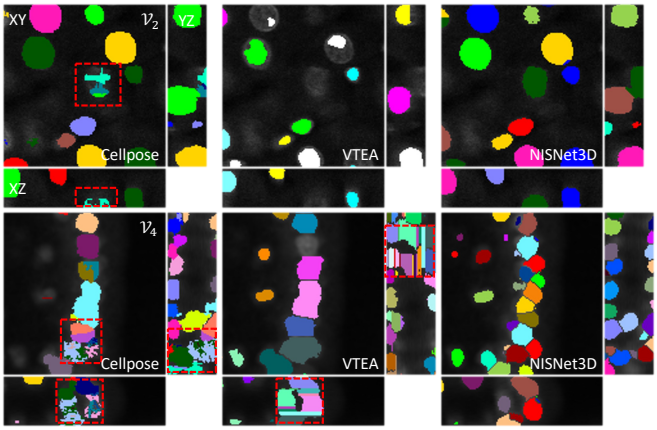

Fig. 15 Visualization of segmentation results of Cellpose and VTEA compared with NISNet3D. The red boxes show the segmentation errors.

Cellpose has trouble capturing some very large or small nuclei in an input subvolume and performs worse on "thinner" subvolumes containing more non-ellipsoidal nuclei. Figure 15 shows the 2D to 3D reconstruction error from Cellpose and VTEA compared with NISNet3D. For StarDist, we observed that

it has difficulty segmenting non-star-convex objects in $\mathcal{V}_3$ and achieves better performance on regular ellipsoidal nuclei in $\mathcal{V}_4$ (See Figure 12).

Figure 12 and Figure 13 are the color coded instance segmentation volumes for compared methods. NISNet3D can accurately identify each individual nucleus and segment the nuclei out from the background structure. Note that NISNet3D does not need any prior information about nuclei size or shape, and does not resize or interpolate the input volume. Using our inference scheme shown in Figure 8, NISNet3D can run on a large volume with any given size without losing accuracy from interpolation. We used object-based evaluation metrics to quantitatively evaluate the performance of NISNet3D and other methods. The summary of evaluation results shown in Table 4 and Table 5 indicate that NISNet3D achieved highest mAP and mF_1 on all of our test datasets. As shown in Figure 11, in order to quantify how well the segmented nuclei matches the ground truth nuclei, we use the Average Precision (AP) under different IoU thresholds criteria and Aggregated Jaccard Index (AJI) to evaluate both segmentation and detection accuracy. Figure 12 and Figure 13 show the color coded instance segmentation results. Our method can better separate the touching nuclei as well as maintaining the nuclei shape.

4 CONCLUSION

In this paper, we described a true 3D Nuclei Instance Segmentation Network, known as NISNet3D, for fluorescence microscopy images analysis. Our approach directly works on 3D volumes by making use of a modified 3D U-Net and a nuclei instance segmentation system for separating touching nuclei based on a 3D vector field volume and a 3D gradient volume. NISNet3D can be trained on both actual microscopy volumes and synthetic microscopy volumes generated using SpCycleGAN or a combination of both. We demonstrate that NISNet3D performs well when compared to other methods on a variety of microscopy data both visually and quantitatively. In addition, we also present three error/difference visualization methods for visualizing segmentation errors in large 3D microscopy volumes without the need of ground truth annotations.

5 ACKNOWLEDGMENT

This research was partially supported by a George M. O'Brien Award from the National Institutes of Health under grant NIH/NIDDK P30 DK079312 and the endowment of the Charles William Harrison Distinguished Professorship at Purdue University.

The authors have no conflicts of interest.

The original image volumes used in this work were provided by Malgorzata Kamocka, Sherry Clendenon, and Michael Ferkowicz at Indiana University.

We gratefully acknowledge their cooperation.

The NISNet3D source code package is available upon request to imart@ecn.purdue.edu. The source code is released under Creative Commons License Attribution-NonCommercial-ShareAlike - CC BY-NC-SA. The source code cannot be used for commercial purposes. The test volumes are also a available from imart@ecn.purdue.edu. The test volumes are released under Creative Commons License Attribution-NonCommercial-NoDerivs - CC BY-NC-ND.

Address all correspondence to Edward J. Delp, ace@ecn.purdue.edu

References

[1] Lucas, A.M., Ryder, P.V., Li, B., Cimini, B.A., Eliceiri, K.W., Carpenter, A.E.: Open-source deep-learning software for bioimage segmentation. Molecular Biology of the Cell **32**(9), 823–829 (2021)

[2] Piccinini, F., Balassa, T., Carbonaro, A., Diosdi, A., Toth, T., Moshkov, N., Tasnadi, E.A., Horvath, P.: Software tools for 3d nuclei segmentation and quantitative analysis in multicellular aggregates. Computational and structural biotechnology journal **18**, 1287–1300 (2020)

[3] Stringer, C., Wang, T., Michaelos, M., Pachitariu, M.: Cellpose: a generalist algorithm for cellular segmentation. Nature Methods **18**(1), 100–106 (2021)

[4] Greenwald, N.F., Miller, G., Moen, E., Kong, A., Kagel, A., Dougherty, T., Fullaway, C.C., McIntosh, B.J., Leow, K.X., Schwartz, M.S., *et al.*: Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. Nature biotechnology **40**(4), 555–565 (2022)

[5] Kromp, F., Fischer, L., Bozsaky, E., Ambros, I.M., Dörr, W., Beiske, K., Ambros, P.F., Hanbury, A., Taschner-Mandl, S.: Evaluation of deep learning architectures for complex immunofluorescence nuclear image segmentation. IEEE Transactions on Medical Imaging **40**(7), 1934–1949 (2021)

[6] Dunn, K.W., Fu, C., Ho, D.J., Lee, S., Han, S., Salama, P., Delp, E.J.: Deepsynth: Three-dimensional nuclear segmentation of biological images using neural networks trained with synthetic data. Scientific Reports **9**(1), 18295–18309 (2019)

[7] Fu, C., Lee, S., Ho, D.J., Han, S., Salama, P., Dunn, K.W., Delp, E.J.: Three dimensional fluorescence microscopy image synthesis and segmentation. Proceedings of the IEEE Conference on Computer Vision and

Pattern Recognition Workshops, 2302–2310 (2018). Salt Lake City, UT

[8] Yushkevich, P.A., Piven, J., Hazlett, H.C., Smith, R.G., Ho, S., Gee, J.C., Gerig, G.: User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. Neuroimage **31**(3), 1116–1128 (2006)

[9] Berger, D.R., Seung, H.S., Lichtman, J.W.: Vast (volume annotation and segmentation tool): efficient manual and semi-automatic labeling of large 3d image stacks. Frontiers in neural circuits **12**, 88 (2018)

[10] Hollandi, R., Diósdi, Á., Hollandi, G., Moshkov, N., Horváth, P.: Annotatorj: an imagej plugin to ease hand annotation of cellular compartments. Molecular biology of the cell **31**(20), 2179–2186 (2020)

[11] Borland, D., McCormick, C.M., Patel, N.K., Krupa, O., Mory, J.T., Beltran, A.A., Farah, T.M., Escobar-Tomlienovich, C.F., Olson, S.S., Kim, M., *et al.*: Segmentor: a tool for manual refinement of 3d microscopy annotations. BMC bioinformatics **22**(1), 1–12 (2021)

[12] Wang, J., Perez, L.: The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 (2017)

[13] Mikołajczyk, A., Grochowski, M.: Data augmentation for improving deep learning in image classification problem. International Interdisciplinary PhD Workshop, 117–122 (2018). Swinoujscie, Poland

[14] Yang, L., Ghosh, R.P., Franklin, J.M., Chen, S., You, C., Narayan, R.R., Melcher, M.L., Liphardt, J.T.: Nuset: A deep learning tool for reliably separating and analyzing crowded cells. PLoS computational biology **16**(9), 1008193 (2020)

[15] Weigert, M., Schmidt, U., Haase, R., Sugawara, K., Myers, G.: Star-convex polyhedra for 3d object detection and segmentation in microscopy. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 3666–3673 (2020). Snowmass, CO

[16] Sadanandan, S.K., Ranefall, P., Le Guyader, S., Wählby, C.: Automated training of deep convolutional neural networks for cell segmentation. Scientific reports **7**(1), 1–7 (2017)

[17] Caicedo, J.C., Roth, J., Goodman, A., Becker, T., Karhohs, K.W., Broisin, M., Molnar, C., McQuin, C., Singh, S., Theis, F.J., *et al.*: Evaluation of deep learning strategies for nucleus segmentation in fluorescence images. Cytometry Part A **95**(9), 952–965 (2019)

[18] Baniukiewicz, P., Lutton, E.J., Collier, S., Bretschneider, T.: Generative adversarial networks for augmenting training data of microscopic cell images. Frontiers in Computer Science, 10 (2019)

[19] Wiesner, D., Nečasová, T., Svoboda, D.: On generative modeling of cell shape using 3d gans, 672–682 (2019)

[20] Chen, A., Wu, L., Han, S., Salama, P., Dunn, K.W., Delp, E.J.: Three dimensional synthetic non-ellipsoidal nuclei volume generation using bezier curves. Proceedings of the IEEE International Symposium on Biomedical Imaging (2021). Nice, France

[21] Wu, L., Han, S., Chen, A., Salama, P., Dunn, K.W., Delp, E.J.: Rcnn-slicenet: A slice and cluster approach for nuclei centroid detection in three-dimensional fluorescence microscopy images. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 3750–3760 (2021). Nashville, TN

[22] Wu, L., Chen, A., Salama, P., Dunn, K.W., Delp, E.J.: An ensemble learning and slice fusion strategy for three-dimensional nuclei instance segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2022). New Orleans, LA

[23] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. International journal of computer vision **128**(2), 261–318 (2020)

[24] Carneiro, G., Zheng, Y., Xing, F., Yang, L.: Review of deep learning methods in mammography, cardiovascular, and microscopy image analysis. Deep Learning and Convolutional Neural Networks for Medical Image Computing, 11–32 (2017)

[25] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention **9351**, 231–241 (2015). Munich, Germany

[26] Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3D u-net: Learning dense volumetric segmentation from sparse annotation. Medical Image Computing and Computer-Assisted Intervention **9901**, 424–432 (2016)

[27] Milletari, F., Navab, N., Ahmadi, S.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. International Conference on 3D Vision, 565–571 (2016)

[28] Balakrishnan, G., Zhao, A., Sabuncu, M.R., Guttag, J., Dalca, A.V.: Voxelmorph: A learning framework for deformable medical image registration. IEEE Transactions on Medical Imaging **38**(8), 1788–1800 (2019)

[29] Graham, S., Vu, Q.D., Raza, S.E.A., Azam, A., Tsang, Y.W., Kwak, J.T., Rajpoot, N.: Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. Medical Image Analysis **58**, 101563 (2019)

[30] Fu, C., Ho, D.J., Han, S., Salama, P., Dunn, K.W., Delp, E.J.: Nuclei segmentation of fluorescence microscopy images using convolutional neural networks. Proceedings of the IEEE International Symposium on Biomedical Imaging, 704–708 (2017). Melbourne, Australia

[31] Ho, D.J., Fu, C., Salama, P., Dunn, K.W., Delp, E.J.: Nuclei segmentation of fluorescence microscopy images using three dimensional convolutional neural networks. IEEE Conference on Computer Vision and Pattern Recognition Workshops, 834–842 (2017)

[32] Schmidt, U., Weigert, M., Broaddus, C., Myers, G.: Cell detection with star-convex polygons. Medical Image Computing and Computer Assisted Intervention, 265–273 (2018). Granada, Spain

[33] Ho, D.J., Montserrat, D.M., Fu, C., Salama, P., Dunn, K.W., Delp, E.J.: Sphere estimation network: three-dimensional nuclei detection of fluorescence microscopy images. Journal of Medical Imaging **7**(4), 1–16 (2020)

[34] Mandal, S., Uhlmann, V.: Splinedist: Automated cell segmentation with spline curves. Proceedings of the International Symposium on Biomedical Imaging, 1082–1086 (2021). Nice, France

[35] Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. Proceedings of the International Conference on Document Analysis and Recognition, 958–963 (2003). Edinburgh, UK

[36] McKinley, S., Levine, M.: Cubic spline interpolation. College of the Redwoods **45**(1), 1049–1060 (1998)

[37] Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. Proceedings of the IEEE International Conference on Computer Vision, 2242–2251 (2017). Venice, Italy

[38] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition,

770–778 (2016). Venice, Italy

[39] Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., Glocker, B., Rueckert, D.: Attention u-net: Learning where to look for the pancreas. arXiv preprint arXiv:1804.03999 (2018)

[40] Han, S., Lee, S., Fu, C., Salama, P., Dunn, K., Delp, E.J.: Nuclei counting in microscopy images with three dimensional generative adversarial networks. Proceedings of the SPIE Conference on Medical Imaging **10949**, 753–763 (2019). International Society for Optics and Photonics. San Diego, CA

[41] Zhang, D., Song, Y., Liu, S., Feng, D., Wang, Y., Cai, W.: Nuclei instance segmentation with dual contour-enhanced adversarial network. Proceedings of the IEEE International Symposium on Biomedical Imaging, 409–412 (2018). Washington, DC

[42] Lin, T., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. Proceedings of the IEEE International Conference on Computer Vision, 2999–3007 (2017). Venice, Italy

[43] Salehi, M., Sadegh, S., Erdogmus, D., Gholipour, A.: Tversky loss function for image segmentation using 3d fully convolutional deep networks. Proceedings of International Workshop on Machine Learning in Medical Imaging, 379–387 (2017). Quebec, Canada

[44] Soille, P., Vincent, L.M.: Determining watersheds in digital pictures via flooding simulations. Visual Communications and Image Processing'90: Fifth in a Series **1360**, 240–250 (1990). International Society for Optics and Photonics

[45] Yang, X., Li, H., Zhou, X.: Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and kalman filter in time-lapse microscopy. IEEE Transactions on Circuits and Systems I: Regular Papers **53**(11), 2405–2414 (2006)

[46] Lin, Z., Wei, D., Petkova, M.D., Wu, Y., Ahmed, Z., Zou, S., Wendt, N., Boulanger-Weill, J., Wang, X., Dhanyasi, N., et al.: Nucmm dataset: 3d neuronal nuclei instance segmentation at sub-cubic millimeter scale. International Conference on Medical Image Computing and Computer-Assisted Intervention, 164–174 (2021)

[47] Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 5967–5976 (2017). Honolulu, HI

[48] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2017)

[49] Lin, T., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. IEEE International Conference on Computer Vision, 2999–3007 (2017). Venice, Italy

[50] Rizk, A., Paul, G., Incardona, P., Bugarski, M., Mansouri, M., Niemann, A., Ziegler, U., Berger, P., Sbalzarini, I.F.: Segmentation and quantification of subcellular structures in fluorescence microscopy images using squassh. Nature Protocols **9**(3), 586–596 (2014)

[51] McQuin, C., Goodman, A., Chernyshev, V., Kamentsky, L., Cimini, B.A., Karhohs, K.W., Doan, M., Ding, L., Rafelski, S.M., Thirstrup, D., Wiegraebe, W., Singh, S., Becker, T., Caicedo, J.C., Carpenter, A.E.: Cellprofiler 3.0: Next-generation image processing for biology. PLoS biology **16**(7), 2005970–117 (2018)

[52] Winfree, S., Khan, S., Micanovic, R., Eadon, M.T., Kelly, K.J., Sutton, T.A., Phillips, C.L., Dunn, K.W., El-Achkar, T.M.: Quantitative three-dimensional tissue cytometry to study kidney tissue and resident immune cells. Journal of the American Society of Nephrology **28**(7), 2108–2118 (2017)

[53] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. Proceedings of the IEEE conference on computer vision and pattern recognition, 658–666 (2019). Long Beach, CA

[54] Powers, D.M.: Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061 (2020)

[55] Hosang, J., Benenson, R., Dollár, P., Schiele, B.: What makes for effective detection proposals? IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(4), 814–830 (2016)

[56] Everingham, M., Eslami, S.M.A., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision **111**(1), 98–136 (2015)

[57] Padilla, R., Netto, S.L., da Silva, E.A.B.: A survey on performance metrics for object-detection algorithms. Proceedings of the International Conference on Systems, Signals and Image Processing, 237–242 (2020). Niteroi, Brazil

[58] Davis, J., Goadrich, M.: The relationship between precision-recall and roc

curves. Proceedings of the international conference on Machine learning, 233–240 (2006). Pittsburgh, PA

[59] Kumar, N., Verma, R., Sharma, S., Bhargava, S., Vahadane, A., Sethi, A.: A dataset and a technique for generalized nuclear segmentation for computational pathology. IEEE Transactions on Medical Imaging **36**(7), 1550–1560 (2017)

[60] Santella, A., Du, Z., Nowotschin, S., Hadjantonakis, A.K., Bao, Z.: A hybrid blob-slice model for accurate and efficient detection of fluorescence labeled nuclei in 3d. BMC bioinformatics **11**(1), 1–13 (2010)