PatchPerPix for Instance Segmentation

Lisa Mais[0000−0002−9281−2668] and Peter Hirsch[0000−0002−2353−5310]
and Dagmar Kainmueller[0000−0002−9830−2415]

Berlin Institute of Health / Max-Delbrueck-Center for Molecular Medicine in the Helmholtz Association, Berlin, Germany
{firstname.lastname}@mdc-berlin.de

Abstract. We present a novel method for proposal free instance segmentation that can handle sophisticated object shapes which span large parts of an image and form dense object clusters with crossovers. Our method is based on predicting dense local shape descriptors, which we assemble to form instances. All instances are assembled simultaneously in one go. To our knowledge, our method is the first non-iterative method that yields instances that are composed of learnt shape patches. We evaluate our method on a diverse range of data domains, where it defines the new state of the art on four benchmarks, namely the ISBI 2012 EM segmentation benchmark, the BBBC010 C. elegans dataset, and 2d as well as 3d fluorescence microscopy data of cell nuclei. We show furthermore that our method also applies to 3d light microscopy data of Drosophila neurons, which exhibit extreme cases of complex shape clusters.

1 Introduction

The task of instance segmentation has a wide range of applications in natural images as well as microscopy images from the biomedical domain. A prevalent class of instance segmentation methods, namely proposal-based methods based on RCNN [10,11], has proven successful in cases where instance location and size can be well-approximated by bounding boxes. However, in many cases, especially in the biomedical domain, this does not hold: Instances may span widely across the image, and hence multiple instances may have very similar, large bounding boxes. To complicate things, instances may be densely clustered, in some cases overlapping, including crossovers. Proposal-free methods are applicable in such cases, where popular choices include metric learning / instance coloring [7,18,4,17], affinity-based methods [8,30,20,9], and learnt watershed [3,31]. However, respective pixel-wise predictions do not explicitly capture instance shape, nor are they suitable for disentangling overlapping instances.

To overcome these limitations, we propose to (1) densely predict representations of the shapes of instance patches, (2) cover the image foreground with the most plausible shape patches, and (3) puzzle together complete instance

Lisa Mais and Peter Hirsch contributed equally, listed in random order.
Code available: https://github.com/Kainmueller-Lab/PatchPerPix

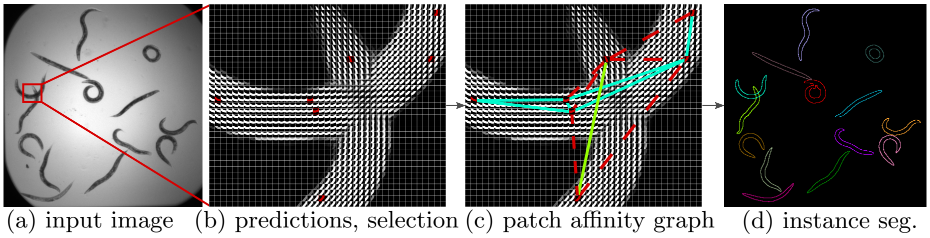


Fig. 1: PatchPerPix overview. Given the raw input image (a), a CNN predicts dense patches for each pixel (b, best seen with zoom) which are then used to find a consensus for each pair of pixels within the patch size. The patches that best agree with this consensus are selected (shown in red in b) and connected to form a patch affinity graph. (c) Edges of the patch affinity graph are assigned scores derived from the agreement of the *merged* shape patches with the consensus. The final instance segmentation (d) is obtained by signed graph partitioning. Shown in (c,d) is the result of connected component analysis on the positive subgraph, where edges with negative scores are depicted in red.

shapes from these patches by means of partitioning a patch affinity graph. The approach of covering the image by selecting from a redundant set of instance patch predictions allows for naturally handling overlap (including crossovers), as overlapping instance patches can be selected, potentially resulting in pixels covered by multiple instances.

Our general idea is closely related to Singling Out Networks [32]. However, they are different in that they rely on a dictionary of known instances, thereby limiting the variability of objects they can handle, and they only consider predicting whole instances and not patches of instances, thereby limiting the size of feasible object categories.

Our shape prediction network predicts, for each pixel of the input image, a representation of the local shape of the instance this pixel belongs to, namely a *shape patch* of the pixel's instance. The architecture we propose is derived from the U-Net [26], thus allowing for efficient dense prediction. As representations of instance patch shapes, we explore local binary masks, as well as encodings (i.e. compressed versions) of these. The idea of predicting instance shape masks per pixel of an image has been pursued before [5,6,15]. However, all these approaches work on the assumption that a shape mask can capture a complete instance shape. Thus they are designed for object categories common to natural images rather than for disentangling clusters of complex shapes that occupy similar bounding boxes, as relevant in the biomedical domain. Predicting shape encodings instead of binary masks is also not new [15]. However, besides only considering complete instance shapes as opposed to our patches of instances, in [15], shape encoding and respective decoder are trained separately, where we show in our work that end-to-end training yields considerable improvement.

The variant of our method that predicts local binary masks as shape representations is closely related to methods that employ long-range affinities [16,30,20,9].

In essence, our predicted binary patches can be interpreted as dense affinities in a neighborhood around each pixel. However, in contrast to affinity-based methods, we instead interpret our predictions as patches of instances, from which we puzzle together complete instances. This way, our yielded global instance shapes are assembled from *learned shape patches*, a property that does not hold for affinity-based methods. Note that in this respect, our method is related to CELIS [23], which learns to agglomerate super-pixels to form instances with plausible shapes, yet their initial pixel-wise predictions do not capture object shape. Furthermore, our method is related to Flood Filling Networks [13], an iterative method that learns to expand instances one-by-one. In contrast, our method segments all instances simultaneously in one pass.

We show in a quantitative evaluation that our method is the new state of the art on the ISBI 2012 challenge on segmentation of neuronal structures in EM stacks [1], outperforms the previous state of the art [32,25,17] on the BBBC010 benchmark dataset of worm images [28] by a large margin, and also outperforms the state of the art [27,29,12] on 2d and 3d light microscopy images of densely packed cell nuclei. Last but not least, we demonstrate that our method also applies to the complex tree-like shapes of neurons in 3d light microscopy images.

In summary, our contributions are:

- A novel method for segmenting instances of complex shapes that spread widely across an image in crowded scenarios, with overlaps and crossovers.
- Instance segmentations are assembled from learnt shape pieces. Our method is, to our knowledge, the first such method that is not iterative, i.e. we compute all instances in one pass.
- Our method defines the new state of the art on the competitive ISBI 2012 EM segmentation challenge, considerably outperforms the state of the art on the challenging BBBC010 C. elegans dataset, and also defines the new state of the art on 2d and 3d benchmark data of cell nuclei.

2 PatchPerPix for Instance Segmentation

We train a CNN to predict dense local shape patches, from which we assemble all instances in an image simultaneously in a one-pass pipeline. Figure 1 and Suppl. Fig. 5 provide an overview of our proposed method, which we term *PatchPerPix*.

Formally, our CNN yields an estimate $p\colon \mathrm{Dom}(I) \times \mathcal{P} \to [0,1]$ of the function

$$\begin{aligned} &p^*\colon \mathrm{Dom}(I) \times \mathcal{P} \to \{0,1\} \\ &\qquad (\mathbf{x}, \mathbf{dx}) \mapsto \begin{cases} 1 \text{ if } \mathrm{Instance}(\mathbf{x}) = \mathrm{Instance}(\mathbf{x}+\mathbf{dx}) \text{ and } \mathbf{x}, \mathbf{x}+\mathbf{dx} \in \mathrm{fg}(I) \\ 0 \text{ otherwise} \end{cases} \end{aligned}$$

that captures, for each pixel $\mathbf{x} \in \mathcal{R}^d$ in the foreground $\mathrm{fg}(I)$ of a d-dimensional image I, and each pixel $\mathbf{x}+\mathbf{dx}$ at a fixed, dense set of offsets $\mathcal{P} \subset \mathcal{R}^d$, whether $\mathbf{x}$ and $\mathbf{x}+\mathbf{dx}$ belong to the same instance.

Section 2.1 describes our proposed instance assembly pipeline given the estimated function p. Section 2.2 describes the CNN architectures we explore to yield p.

2.1 Instance Assembly

We denote a restriction of the estimated function p to a single pixel as

$$p_{\mathbf{x}} \colon \mathbf{x} + \mathcal{P} \rightarrow [0,1]\,,\ \mathbf{y} \mapsto p(\mathbf{x}, \mathbf{y} - \mathbf{x})$$

We denote the domain of $p_{\mathbf{x}}$ as $\mathrm{patch}(p_{\mathbf{x}}) := \mathbf{x} + \mathcal{P}$. For each patch, the pixels that are predicted to belong to the same instance as $\mathbf{x}$ by means of a probability threshold t, i.e. the pixels classified as foreground w.r.t. the instance at $\mathbf{x}$, are denoted as

$$\mathrm{fg}(p_{\mathbf{x}}) := \{\mathbf{y} \in \mathrm{patch}(p_{\mathbf{x}}) : p_{\mathbf{x}}(\mathbf{y}) > t\}\,, \tag{1}$$

and, accordingly, the respective background pixels as

$$\mathrm{bg}(p_{\mathbf{x}}) := \{\mathbf{y} \in \mathrm{patch}(p_{\mathbf{x}}) : p_{\mathbf{x}}(\mathbf{y}) < 1 - t\}\,. \tag{2}$$

For each pixel pair $(\mathbf{y}, \mathbf{z})$ covered by at least one informative patch, i.e. $\exists \mathbf{x} \in \mathrm{Dom}(I) : \{\mathbf{y}, \mathbf{z}\} \subset \mathrm{patch}(p_{\mathbf{x}}) \wedge \{\mathbf{y}, \mathbf{z}\} \cap \mathrm{fg}(p_{\mathbf{x}}) \neq \emptyset$, summing up observations from all patches yields a consensus that $\mathbf{y}$ and $\mathbf{z}$ belong to the same instance, i.e. a consensus affinity

$$\begin{aligned} \mathrm{aff}(\mathbf{y}, \mathbf{z}) := \frac{1}{Z_{\mathrm{aff}}(\mathbf{y}, \mathbf{z})} \cdot \Bigg(& \sum_{\substack{\mathbf{x} \in \mathrm{Dom}(I): \\ \{\mathbf{y},\mathbf{z}\} \subset \mathrm{fg}(p_{\mathbf{x}})}} p_{\mathbf{x}}(\mathbf{y}) \cdot p_{\mathbf{x}}(\mathbf{z}) \\ & - \sum_{\substack{\mathbf{x} \in \mathrm{Dom}(I): \\ \mathbf{y} \in \mathrm{fg}(p_{\mathbf{x}}), \mathbf{z} \in \mathrm{bg}(p_{\mathbf{x}})}} p_{\mathbf{x}}(\mathbf{y}) \cdot (1 - p_{\mathbf{x}}(\mathbf{z})) - \sum_{\substack{\mathbf{x} \in \mathrm{Dom}(I): \\ \mathbf{y} \in \mathrm{bg}(p_{\mathbf{x}}), \mathbf{z} \in \mathrm{fg}(p_{\mathbf{x}})}} (1 - p_{\mathbf{x}}(\mathbf{y})) \cdot p_{\mathbf{x}}(\mathbf{z}) \Bigg) \end{aligned} \tag{3}$$

with normalization factor

$$Z_{\mathrm{aff}}(\mathbf{y}, \mathbf{z}) := |\{\mathbf{x} \in \mathrm{Dom}(I) : \{\mathbf{y}, \mathbf{z}\} \subset \mathrm{patch}(p_{\mathbf{x}}) \wedge \{\mathbf{y}, \mathbf{z}\} \cap \mathrm{fg}(p_{\mathbf{x}}) \neq \emptyset\}|. \tag{4}$$

Given these consensus affinities, we define a score for each patch with non-empty foreground by assessing how well it agrees with the consensus:

$$\mathrm{score}(p_{\mathbf{x}}) := \frac{1}{Z_{\mathrm{score}}(p_{\mathbf{x}})} \cdot \Bigg(\sum_{\{\mathbf{y},\mathbf{z}\} \subset \mathrm{fg}(p_{\mathbf{x}})} \mathrm{aff}(\mathbf{y}, \mathbf{z}) - \sum_{\substack{\mathbf{y} \in \mathrm{fg}(p_{\mathbf{x}}), \\ \mathbf{z} \in \mathrm{bg}(p_{\mathbf{x}})}} \mathrm{aff}(\mathbf{y}, \mathbf{z}) \Bigg) \tag{5}$$

with normalization factor

$$Z_{\mathrm{score}}(p_{\mathbf{x}}) := |\{\{\mathbf{y}, \mathbf{z}\} \subset \mathrm{patch}(p_{\mathbf{x}}) : \{\mathbf{y}, \mathbf{z}\} \cap \mathrm{fg}(p_{\mathbf{x}}) \neq \emptyset\}|.$$

We rank all patches w.r.t. their score (Eq. 5). We employ a greedy set cover algorithm to select high-ranking patches whose patch foregrounds $\mathrm{fg}(p_{\mathbf{x}})$ fully cover the image foreground $\mathrm{fg}(I)$. Section 2.2 describes how we obtain the image foreground. In more detail, the set cover algorithm proceeds as follows: Iterating from high to low score over the ranked list of patches, we pre-select patches if they cover previously uncovered image foreground, until the image foreground is fully covered. We further thin out this pre-selection as follows: We iteratively select as next patch from the pre-selection the patch that covers the most remaining foreground, until the whole foreground is covered.

Given this selection of high-ranking patches, the consensus affinities (Eq. 3) allow us to define a score that measures for a *pair of patches* whether they belong to the same instance, i.e. a consensus affinity between $p_{\mathbf{x}}$ and $p_{\mathbf{y}}$:

$$\operatorname{paff}(p_{\mathbf{x}}, p_{\mathbf{y}}) := \frac{1}{Z_{\operatorname{paff}}(p_{\mathbf{x}}, p_{\mathbf{y}})} \cdot \sum_{\substack{\mathbf{v} \in \operatorname{fg}(p_{\mathbf{x}}),\\ \mathbf{w} \in \operatorname{fg}(p_{\mathbf{y}})}} \operatorname{aff}(\mathbf{v}, \mathbf{w}) \tag{6}$$

with normalization factor

$$\begin{aligned} Z_{\operatorname{paff}}(p_{\mathbf{x}}, p_{\mathbf{y}}) := |\{\mathbf{v} \in \operatorname{fg}(p_{\mathbf{x}}), \mathbf{w} \in \operatorname{fg}(p_{\mathbf{y}}) :\\ \exists \mathbf{z} : \{\mathbf{v}, \mathbf{w}\} \subset \operatorname{patch}(p_{\mathbf{z}}) \wedge \{\mathbf{v}, \mathbf{w}\} \cap \operatorname{fg}(p_{\mathbf{z}}) \neq \emptyset\}|. \end{aligned}$$

We compute patch pair affinities (Eq. 6) between selected high-ranking patches iff the respective $Z_{\operatorname{paff}}(\cdot, \cdot) > 0$, yielding a patch affinity graph. We partition this graph via connected component analysis on the positive subgraph, or alternatively by means of the mutex watershed algorithm [30], depending on the application domain. We obtain the final instance segmentation by assigning, per connected component, a unique instance ID to all pixels contained in the union of the respective patch foregrounds. Note that in general, this may assign multiple instance IDs to some pixels, which is desired in some, but not all, applications. In case overlapping instances are not desired, we assign the ID of the patch prediction with highest probability at the respective pixel.

We implemented the computationally expensive parts of our instance assembly pipeline in CUDA for efficient execution. In applications with sparse image foreground, we further improve computational efficiency by restricting $\operatorname{patch}(p_{\mathbf{x}})$ to the image foreground, i.e. $\operatorname{patch}_{\text{sparse}}(p_{\mathbf{x}}) := \operatorname{patch}(p_{\mathbf{x}}) \cap \operatorname{fg}(I)$.

2.2 CNN Architecture

We train a deep convolutional neural network to predict the function p. It does so by predicting $p_{\mathbf{x}}(\mathbf{x}+\mathcal{P})$ for each pixel of the input image. Thus the cardinality of the set $\mathcal{P}$ determines the number of output channels of the network. We train the network w.r.t. standard cross-entropy loss averaged over all outputs. We use a U-Net [26] as backbone architecture. To facilitate predictions of shape representations with hundreds of dimensions, we keep the number of feature maps fixed (instead of reducing) in the upward path of the U-Net. Thus we avoid having to predict high-dimensional pixel-wise outputs from only tens of feature maps as present in the penultimate layer of a standard U-Net.

Our baseline PatchPerPix architecture, termed **ppp**, is a U-Net that directly outputs $p_{\mathbf{x}}$ at each pixel $\mathbf{x}$ of the input image I. To estimate the image foreground $\operatorname{fg}(I)$, we include offset $\mathbf{0}$ in $\mathcal{P}$. A practical issue with ppp is that the size of the predicted patches, i.e. the number of outputs of the U-Net, is limited by GPU

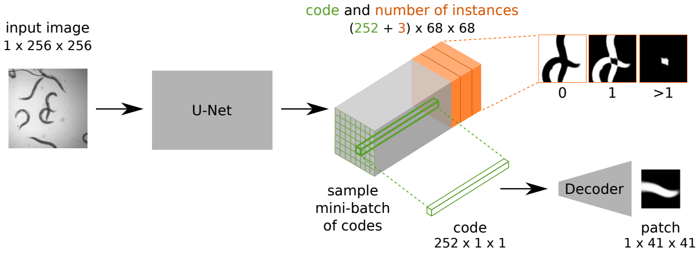


Fig. 2: **ppp+dec architecture:** A U-Net predicts shape patch encodings, which are fed into the decoding path of an auto-encoder. Additional outputs of the U-Net predict the number of instances at each pixel. U-Net and decoder are trained jointly end-to-end. Categorical cross-entropy is used for the number of instances, and binary cross-entropy for the patch predictions. Both losses are summed up without weighting. The batch of codes that is run through the decoder is sampled from pixels for which the number of instances is predicted to be 1.

memory. Furthermore, in most application domains, the variety of possible patch predictions, and hence the amount of information contained in each, is limited. Therefore, in addition to our baseline model, we explore two variants that learn compressed representations of $p_{\mathbf{x}}$ and decode these via (1) the decoder part of a separately trained autoencoder (ppp+ae), and (2) a decoder that is trained end-to-end with the backbone U-Net (ppp+dec), as described in the following.

ppp+ae. In a separate first step, we train a fully convolutional autoencoder on patches of ground truth binary masks to learn a patch latent space. The backbone U-Net is then trained to regress a respective learnt latent vector (a.k.a. ”encoding” or ”code”) for each pixel of the input image w.r.t. sum of squared differences loss. To de-compress patch predictions for our instance assembly pipeline, the decoder part of the pre-trained autoencoder is employed. We add an extra output channel to the U-Net to predict a foreground mask, trained w.r.t. cross-entropy loss and added to the code loss without any weighting. Codes are decoded for all foreground pixels obtained by thresholding the foreground mask.

ppp+dec. Here, we attach the decoder part of the autoencoder used in ppp+ae to the end of the U-Net and train the resulting joint network end-to-end from scratch w.r.t. cross-entropy. As before, the U-Net part of the network outputs the code. However, there is no loss employed directly on the code. To fit end-to-end training onto GPU memory, we sample codes from ground truth foreground pixels at training time, which are then fed to the decoder network. Similarly to ppp+ae, we extend the U-Net to simultaneously predict the foreground to allow for decoding only foreground pixels. This architecture is depicted in Figure 2.

ppp+dec combines a U-Net for predicting a shape encoding with the decoder part of an auto-encoder. Interestingly, this end-to-end trainable architecture has two decoding parts, namely (1) the upward path of the U-Net, which serves for combining high-level image information captured at lower layers with low-level information from upper layers, and (2) the decoder half of an auto-encoder,

which is needed to decompress local shape predictions in the end. Furthermore, especially when dealing with sparse data, the U-Net performs many dispensable computations, namely on background pixels. Hence we investigated whether our proposed architecture could be replaced by a standard encoder-decoder architecture alone, with one encoding and one decoding path, like e.g. [2], as follows:
ed-ppp. Our architecture takes an image patch the size of our shape patches plus some surrounding receptive field as input, and generates a shape patch for the respective central pixel's instance as output. It is applied in a sliding window fashion on all pixels in the image foreground. We use 3×3 down- and upsampling to facilitate the singling out of the center pixel's instance from its neighboring instances. To determine for which pixels to run the encoder-decoder network, we train a separate U-Net to generate a foreground mask in a preceding step.

2.3 Overlapping Regions

The case of multiple objects sharing pixels can be found in many biomedical applications, e.g. in 2d images of model organisms such as worms that crawl on top of each other, or neurons in light microscopy data that share pixels due to the partial volume effect. As pixels located in areas of overlap belong to multiple instances, their respective shape patch is not well-defined. Hence we exclude these pixels from the entire pipeline. During training, we achieve this by masking out these areas in the loss computation. To detect overlap at test time, we predict the number of instances per pixel by extending the foreground classification task by an ”overlap” class, which, as before, is trained jointly with the patch predictions by means of added cross-entropy loss. This information is then used in the instance assembly: Pixels in overlapping regions are discarded, i.e. their respective shape patches do not contribute to the consensus and cannot be selected. This constitutes a limitation of our method in that (i) only overlapping regions with a maximum diameter of smaller than the size of the patches can be covered completely by patch shapes, and (ii) only occlusions within the range of the neighborhood used in the patch graph generation can be bridged.

3 Results

We evaluate our method on four benchmark datasets, which comprise overlapping objects, sophisticated object shapes, and, to show the generic applicability of our method, also simple object shapes. The first dataset, the BBBC010 C. elegans dataset [21], exhibits clusters of overlapping objects with large, coinciding bounding boxes. The second dataset, the ISBI 2012 Challenge on segmenting neuronal structures in electron microscopy [1], exhibits densely clustered objects with sophisticated shapes that span the whole image, albeit without overlaps. The third and fourth dataset exhibit densely clustered objects of simple, approximately ellipsoidal shapes, namely 2d and 3d fluorescence light microscopy datasets of cell nuclei [27,29]. Our results define the new state of the art in all cases, as detailed in Sections 3.1, 3.2, and 3.3. Furthermore, we study the impact of individual steps of our instance assembly pipeline as well as our proposed network architecture designs on BBBC010. Last, we show promising qualitative results on 3d light microscopy data of neurons, which exhibit extreme cases of sophisticated object shapes that form dense clusters with overlaps (Sec. 3.4).

3.1 BBBC010 C. elegans worm disentanglement

The **BBBC010** dataset from the Broad Bioimage Benchmark Collection [21][1] consists of 100 brightfield microscopy images showing multiple C. elegans worms per image, which may overlap and cluster. As ground truth, to capture overlaps correctly, BBBC010 provides an individual binary mask for each worm.

In this Section, we report a quantitative evaluation of our method in comparison with related work [32,25,17]. Furthermore, we report a comparison of the neural network architecture designs we explored, as well as an ablation study that assesses the impact of individual steps of our instance assembly pipeline. Patch- and code-size hyperparameters are studied in the Supplement. We report results in terms of the AP_{dsb} metric used in the kaggle 2018 data science bowl, which takes both missing and spurious instances into account[2]. We also report a range of additional metrics that have been reported for competing approaches, including the slightly different AP_{COCO}, thus enabling direct comparability.

As backbone CNN architecture we employ a 4-level U-Net [26] starting with 40 feature maps, with two-fold down- and upsampling operations, and constant number of feature maps during upsampling. Our network takes raw brightfield images as sole input, while [25,32,17] additionally exploit ground truth segmentations of the image foreground as input. In the ppp architecture, we employ a patch size of 25×25, yielding a U-Net with 625 outputs. In the ppp+ae and ppp+dec architectures, we employ a code of size 252 as intermediate output of the U-Net, which is then fed into a decoder network to yield a patch of size 41×41. The ed-ppp architecture takes 81×81 patches of the raw image as input and predicts 41×41 shape patches. It applies 3×3 max-pooling three times, and has two convolutional layers on each level. At the bottleneck, the code has an extent of $3 \times 3 \times 256$ and uses 1×1 convolutions. The network is symmetric and uses same padding. The output is cropped to obtain the desired patch shape.

As in related work [32,25,17], we divide the BBBC010 dataset into training- and test set with 50 images each. We apply 2-fold cross-validation on the test set to determine the number of training steps and the patch foreground threshold t (Eq. 1), individually in all experiments. For training, we use standard augmentation including elastic deformations in all experiments. Contrary to [32], we do not augment the number of worms synthetically, but focus on crowded regions during training. For patch graph partitioning, we explore connected component analysis on the positive subgraph (CC) as well as the mutex watershed (MWS) [30]. In our result tables, MWS is the default if not noted otherwise.

[1] BBBC010v1: C.elegans infection live/dead image set version 1 provided by Fred Ausubel

[2] https://www.kaggle.com/c/data-science-bowl-2018

Table 1: Quantitative results on the BBBC010 dataset. Top: We compare to competing approaches in various metrics due to a missing standard: [25,17] report COCO metrics [19], [32] plot the recall for different thresholds, [28] evaluate the percentage of ground truth worms which are matched with pixelwise F1 score above 0.8. Bottom: Results for the architecture setups we explored.

BBBC010						
AP_{COCO}	$avAP_{[0.5:0.05:0.95]}$	$AP_{0.5}$	$AP_{0.75}$	$Recall_{0.5}$	$Recall_{0.8}$	$F1_{0.8}$
Semi-conv Ops [25]	0.569	0.885	0.661	-	-	-
SON [32]	-	-	-	∼ 0.97	∼ 0.7	-
WormToolbox [28]	-	-	-	-	-	0.81
Harmonic Emb. [17]	0.724	0.900	0.723	-	-	-
PatchPerPix (ppp+dec)	**0.775**	**0.939**	**0.891**	**0.987**	**0.895**	**0.978**
AP_{dsb}	$avAP_{[0.5:0.05:0.95]}$	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$
ppp	0.689	0.890	0.872	0.840	0.710	0.372
ppp+ae	0.617	0.878	0.831	0.783	0.610	0.199
ppp+dec	**0.727**	**0.930**	**0.905**	**0.879**	**0.792**	**0.386**
ed-ppp	0.675	0.891	0.853	0.820	0.734	0.309

Table 1 compares state-of-the-art methods [25,32,17] and PatchPerPix variants. Table 2 lists results of our ablation study. Figure 3 shows exemplary PatchPerPix results for different CNN architectures. Suppl. Fig. 1 compares PatchPerPix with Singling Out Networks [32] on an exemplary image.

PatchPerPix improves over competing methods by a considerable margin (cf. Table 1, top). Singling Out Networks (SON [32]) are of limited pixel accuracy by design, hence superior performance of PatchPerPix at high IoU thresholds is no surprise. However, PatchPerPix is not just more pixel accurate, but outperforms SON across the IoU threshold range. PatchPerPix also outperforms Harmonic Embeddings [17], a metric learning variant that amends the restricted pixel accuracy of SON, yet struggles at disentangling dense clusters of worms.

Interestingly, ppp+dec does not just outperform the separately trained ppp+ae, but also outperforms ed+ppp. I.e., *using a full U-Net as an encoder,* followed by a standard decoder, considerably outperforms a standard encoder-decoder architecture applied in a sliding-window fashion (cf. Table 1, bottom).

Our ablation study (Table 2) shows the significant impact of two core ideas of our instance assembly pipeline, namely consensus affinity computation (absent in MWS-Dense, avAP -0.176) and selecting a sparse set of high-ranking patch predictions while weeding out low-ranking ones by means of consensus agreement scores (absent in ppp+dec w/o selection, avAP - 0.131). These scores correlate significantly with true patch quality (cf. Suppl. Fig. 4). Thinning out a preselection of high ranking patches has a small impact on accuracy (ppp+dec w/o thinout, avAP - 0.004), yet also positively affects run-time. Patch graph partitioning via CC vs. MWS are on a par on the BBBC010 data.

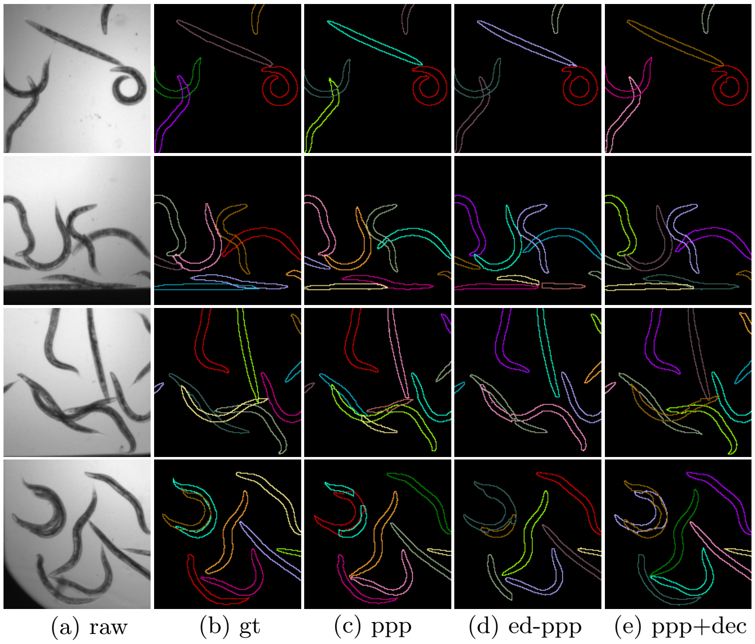


Fig. 3: Qualitative results for exemplary challenging regions of the BBBC010 dataset. All architectures are able to handle crowded and overlapping regions, where ppp+dec yields fewest errors. However, rare shapes such as very bent worms are segmented with slightly higher accuracy by ppp.

Suppl. Fig. 2 shows exemplary failure cases of ppp+dec. Interestingly, strongly bent worms are captured with inferior pixel accuracy by our encoding-based model ppp+dec as opposed to ppp (see Suppl. Fig. 2 right and Fig. 3 top row).

3.2 ISBI 2012 neuron EM segmentation

We evaluate our method on the ISBI 2012 Challenge on segmenting neuronal structures in electron microscopy (EM) data [1]. The data consists of 30 slices of 512x512 pixels with known ground truth (training data), and another 30 such slices for which ground truth is kept secret by the Challenge organizers (test data). Our network architecture as well as the training- and prediction procedure closely follows [30], with the difference that our network has 625 instead of 17 outputs, namely patches of size 1x25x25, and we do not reduce the number of filters in the upward path of the U-Net. For partitioning the patch graph, we use the mutex watershed algorithm [30], which has proven powerful in avoiding false mergers in case of missing neuron membrane signal in the image data.

Our method is the leading entry on the Challenge's leaderboard[3] at present among thousands of submissions by more than 200 teams. Table 3 lists results

[3] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new

Table 2: Ablation study for PatchPerPix on the BBBC010 dataset. We ablate consensus affinity computation as a whole by running graph partitioning directly on the predictions p interpreted as dense affinities (MWS-Dense). We ablate patch selection as a whole (ppp+dec w/o selection), and thinning of the patch selection (ppp+dec w/o thinout). We run ppp+dec with a standard U-Net, i.e. with decreasing number of feature maps in the up-sampling path (ppp+dec std U-Net). Last, we compare patch graph partitioning with CC vs. MWS.

AP_{dsb}	$avAP_{[0.5:0.05:0.95]}$	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$
MWS-Dense	0.551	0.687	0.676	0.661	0.586	0.326
ppp+dec w/o selection	0.596	0.878	0.853	0.798	0.544	0.157
ppp+dec w/o thinout	0.723	0.924	0.898	0.871	0.788	0.393
ppp+dec std U-Net	0.719	0.916	0.891	0.873	0.766	**0.406**
ppp+dec, CC	0.723	0.922	0.894	0.873	0.780	**0.406**
ppp+dec, MWS	**0.727**	**0.930**	**0.905**	**0.879**	**0.792**	0.386

obtained with PatchPerPix in terms of the Challenge error metrics, robust Rand score (rRAND) and robust information theoretic measure (rINF), evaluated on the test data. For comparison, the table also lists the previous state of the art as obtained via sparse affinity predictions processed with the mutex watershed algorithm [30] (MWS). Furthermore, as an additional baseline, we interpreted our patch predictions as dense affinities which we processed with the mutex watershed algorithm as in [30] (MWS-Dense). PatchPerPix slightly outperforms MWS in terms of the leaderboard-defining rRAND score. This can be attributed to fewer mistakes on large neuronal bodies which have respective large impact on the rRAND score. However, the number of such large mistakes we were able to identify by eye on the test set is very small in both approaches.

Interestingly, MWS-Dense performs considerably worse than both PatchPerPix and MWS. The difference between MWS-Dense and PatchPerPix can be attributed to individual erroneous predictions causing errors in MWS-Dense, which are amended in PatchPerPix by our proposed consensus voting and patch selection scheme. As for the difference between MWS-Dense and MWS, we hypothesize that this is due to MWS smartly distinguishing between purely attractive short-range- and purely repulsive long-range affinities. Instead, MWS-Dense treats all affinities as both attractive and repulsive.

3.3 Nuclei segmentation in 2d and 3d

We evaluate our method on 2d and 3d fluorescence microscopy images of cell nuclei. The 2d dataset is a subset of the kaggle 2018 data science bowl[4] as defined in [27]. It consists of 380 training, 67 validation and 50 test images. We refer to this dataset as **dsb2018**. The 3d dataset consists of 28 confocal microscopy images collected and annotated by [22]. Image size is approximately

[4] BBBC038v1: available from the Broad Bioimage Benchmark Collection[21]

Table 3: Quantitative results for the ISBI 2012 Challenge on segmenting neuronal structures in electron microscopy data [1]. PatchPerPix defines the current state-of-the-art in terms of the leaderboard-defining rRAND score.

ISBI2012	rRAND	rINF
PatchPerPix	**0.988290**	0.991544
MWS [30]	0.987922	**0.991833**
MWS-Dense	0.979112	0.989625

140 × 140 × 1100 pixels. Each image shows hundreds of nuclei, with multiple dense clusters. An example is shown in Suppl. Fig. 3. We partition the data as in [29,12], with 18 images for training, 3 for validation, and 7 for testing. We refer to this dataset as **nuclei3d**.

For dsb2018, our CNN architecture is a 4-level U-Net, with 40 initial feature maps, that predicts foreground/background labels as well as codes of size 256, decoded into patches of size 25×25. We determine the number of training steps as well as the patch threshold on the validation set. For nuclei3d, we employ a 3-level 3d U-Net with 20 initial feature maps, tripled after each downsampling step. We predict patches of size $9 \times 9 \times 9$. We filter out instances smaller than a threshold. We determine the number of training steps, the patch threshold, and the instance size threshold on the validation set.

Table 4 lists our results in comparison to the previous state of the art on this data [27,29,12]. We furthermore compare to MALA [8], an affinity-based instance segmentation method trained with a structured loss, which is an established baseline for a different kind of 3d data, namely 3d electron microscopy of neuronal structures, but does not explicitly capture instance shape. For MALA, we employ the same backbone U-Net as for PatchPerPix for a fair comparison.

Superior avAP of PatchPerPix compared to [27,29] can be attributed to superior performance at high IoU thresholds, where StarDist's pixel accuracy is limited due to its coarse polyhedral shape representation, especially in 3d. We list IoU thresholds down to 0.1 as in [29], indicating that PatchPerPix is on a par with StarDist in terms of topological segmentation errors like false splits and false mergers of nuclei. Compared to a recently proposed 3-label U-Net trained with an auxiliary task [12], again, the high pixel accuracy of PatchPerPix leads to slightly higher avAP, while [12] is slightly superior at low IoU thresholds.

On dsb2018, we observed a similar improvement of ppp+dec over ppp as on BBBC010. However, this does not hold for nuclei3d, where ppp+dec did not improve over ppp. We hypothesize that encodings are less able to capture the ellipsoidal shape of nuclei at the very small 3d patch size of 9x9x9 we're bound to to achieve manageable computational performance of instance assembly in 3d (see Suppl. Table 4 for run-times). This performance bottleneck constitutes a current limitation of our method on 3d data, and is subject to future work.

Table 4: Quantitative results for the nuclei datasets dsb2018 and nuclei3d. We report average precision (AP_{dsb}) for multiple IoU thresholds.

AP_{dsb}	avAP	$\text{AP}_{0.1}$	$\text{AP}_{0.2}$	$\text{AP}_{0.3}$	$\text{AP}_{0.4}$	$\text{AP}_{0.5}$	$\text{AP}_{0.6}$	$\text{AP}_{0.7}$	$\text{AP}_{0.8}$	$\text{AP}_{0.9}$
	[0.5:0.1:0.9]									
					dsb2018					
Mask R-CNN[27]	0.594	-	-	-	-	0.832	0.773	0.684	0.489	0.189
StarDist[27]	0.584	-	-	-	-	0.864	0.804	0.685	0.450	0.119
PatchPerPix	**0.693**	**0.919**	**0.919**	**0.915**	**0.898**	**0.868**	**0.827**	**0.755**	**0.635**	**0.379**
					nuclei3d					
MALA [8]	0.381	0.895	0.887	0.859	0.803	0.699	0.605	0.424	0.166	0.012
StarDist 3D[29]	0.406	0.936	0.926	0.905	**0.855**	0.765	0.647	0.460	0.154	0.004
3-label+cpv[12]	0.425	**0.937**	**0.930**	**0.907**	0.848	0.750	0.641	0.473	0.224	**0.035**
PatchPerPix	**0.436**	0.926	0.918	0.900	0.853	**0.766**	**0.668**	**0.493**	**0.228**	0.027

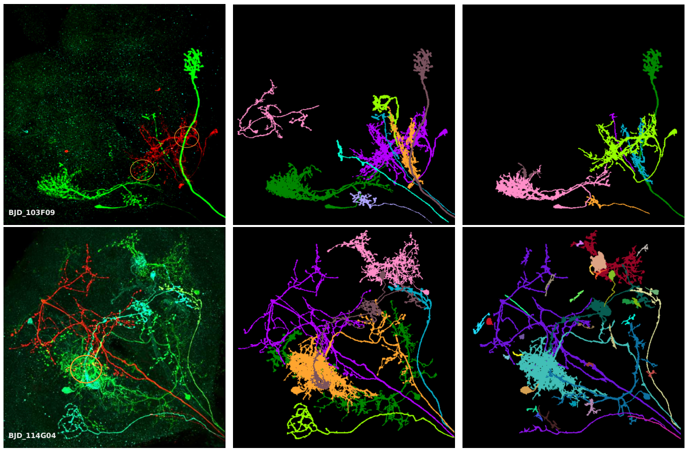
3.4 Neuron separation in 3d light microscopy data

We aim to identify and segment neurons of the fruit fly brain (GAL4 lines [14]) in an unpublished dataset of 3d multicolor confocal microscopy images. The imaging is done by stochastic labeling able to express different densities of neurons [24] (cf. Fig. 4a). This instance segmentation task is very challenging as the number of neurons can be high and image quality is bounded by the necessity to perform large-scale imaging. Moreover, the neurons are very thin, tree-like structures which are intertwined and may overlap due to partial volume effects.

As this dataset is still in the process of being curated and extended, and no competing approach has yet been reported, we do not perform a quantitative evaluation of PatchPerPix, but show the quality of exemplary results on a test set of two images in Figure 4. We use a 3-level 3d U-Net with 2× down- and upsampling and 12 initial feature maps, tripled at each downsampling. The predicted patches are of size $7 \times 7 \times 7$ pixels. Our results serve as proof-of-concept that our method is applicable and yields reasonable results for thin, complex tree-like structures in large 3d image volumes.

4 Conclusion

In this work we present a novel generic method for instance segmentation that comprises a CNN to predict dense local shape descriptors and a one-pass instance assembly pipeline. The method is able to handle objects of sophisticated shapes that appear in dense clusters with overlaps, including crossovers. It is the first to assemble all instances from learnt shape patches, simultaneously in one pass. We successfully applied our method to a range of domains, showing that it (1) outperforms the state of the art on the heavily contested ISBI 2012 challenge on neuron segmentation in electron microscopy, (2) outperforms the state of



(a) Raw (b) Ground Truth (c) PatchPerPix

Fig. 4: Qualitative results on 3d neuron light microscopy examples. (a) Maximum intensity projection of raw images. Orange circles indicate overlapping areas in 3d. Ground truth data (b) were generated by manual segmentation using VVD Viewer. PatchPerPix (c) shows promising results on this challenging dataset.

the art on the challenging BBBC010 C. elegans worm data by a large margin, (3) outperforms the state of the art on 2d and 3d fluorescence microscopy data of densely clustered cell nuclei (on par in terms of cell detection performance, better in terms of pixel accuracy), showing that our method performs well also for simple (blob-like) instance shapes, and (4) can be applied to extreme cases of instance shapes, like neurons in 3d fluorescence microscopy. Future work will tackle a performance bottleneck that becomes relevant on 3d data, where we're currently restricted to patch sizes that are most probably sub-optimally small.

Acknowledgments. We wish to thank Constantin Pape for his invaluable help in reproducing the training- and prediction setup from [30], Carolina Waehlby for help with the BBBC010 data, Stephan Saalfeld and Carsten Rother for inspiring discussions, the FlyLight Project Team[5] at Janelia Research Campus for providing unpublished data, and Claire Managan and Ramya Kappagantula (Janelia Project Technical Resources) for their conscientious manual neuron segmentations. P.H., L.M. and D.K. were funded by the Berlin Institute of Health and the Max Delbrueck Center for Molecular Medicine. P.H. was funded by HFSP grant RGP0021/2018-102. P.H., L.M. and D.K. were supported by the HHMI Janelia Visiting Scientist Program. VVD Viewer[6] is an open-source software funded by NIH grant R01-GM098151-01.

[5] https://www.janelia.org/project-team/flylight

[6] https://github.com/takashi310/VVD_Viewer

References

1. Arganda-Carreras, I., Turaga, S.C., Berger, D.R., Cireşan, D., Giusti, A., Gambardella, L.M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J.M., et al.: Crowdsourcing the creation of image segmentation algorithms for connectomics. Frontiers in neuroanatomy **9**, 142 (2015) 3, 7, 10, 12
2. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence **39**(12), 2481–2495 (2017) 7
3. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. CoRR **abs/1611.08303** (2016) 1
4. Chen, L., Strauch, M., Merhof, D.: Instance segmentation of biomedical images with an object-aware embedding learned with local constraints. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 451–459. Springer (2019) 1
5. Chen, X., Girshick, R.B., He, K., Dollár, P.: Tensormask: A foundation for dense object segmentation. CoRR **abs/1903.12174** (2019), http://arxiv.org/abs/1903.12174 2
6. Dai, J., He, K., Li, Y., Ren, S., Sun, J.: Instance-sensitive fully convolutional networks. CoRR **abs/1603.08678** (2016), http://arxiv.org/abs/1603.08678 2
7. De Brabandere, B., Neven, D., Van Gool, L.: Semantic instance segmentation with a discriminative loss function. arXiv preprint arXiv:1708.02551 (2017) 1
8. Funke, J., Tschopp, F., Grisaitis, W., Sheridan, A., Singh, C., Saalfeld, S., Turaga, S.: Large scale image segmentation with structured loss based deep learning for connectome reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence **PP**, 1–1 (05 2018). https://doi.org/10.1109/TPAMI.2018.2835450 1, 12, 13
9. Gao, N., Shan, Y., Wang, Y., Zhao, X., Yu, Y., Yang, M., Huang, K.: Ssap: Single-shot instance segmentation with affinity pyramid. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 642–651 (2019) 1, 2
10. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 580–587. CVPR '14, IEEE Computer Society, Washington, DC, USA (2014). https://doi.org/10.1109/CVPR.2014.81, https://doi.org/10.1109/CVPR.2014.81 1
11. He, K., Gkioxari, G., Dollr, P., Girshick, R.: Mask r-cnn (2017), http://arxiv.org/abs/1703.06870, cite arxiv:1703.06870Comment: open source; appendix on more results 1
12. Hirsch, P., Kainmueller, D.: An auxiliary task for learning nuclei segmentation in 3d microscopy images. In: Medical Imaging with Deep Learning (MIDL) (July 2020) 3, 12, 13
13. Januszewski, M., Maitin-Shepard, J., Li, P., Kornfeld, J., Denk, W., Jain, V.: Flood-filling networks. CoRR **abs/1611.00421** (2016), http://arxiv.org/abs/1611.00421 3
14. Jenett, A., Rubin, G.M., Ngo, T.T.B., Shepherd, D., Murphy, C., Dionne, H., Pfeiffer, B.D., Cavallaro, A., Hall, D., Jeter, J., Iyer, N., Fetter, D., Hausenfluck, J.H., Peng, H., Trautman, E.T., Svirskas, R.R., Myers, E.W., Iwinski, Z.R., Aso, Y., DePasquale, G.M., Enos, A., Hulamm, P., Lam, S.C.B., Li, H.H., Laverty, T.R., Long, F., Qu, L., Murphy, S.D., Rokicki, K., Safford, T.,

Shaw, K., Simpson, J.H., Sowell, A., Tae, S., Yu, Y., Zugates, C.T.: A gal4-driver line resource for drosophila neurobiology. Cell reports **2**(4), 991–1001 (Oct 2012). https://doi.org/10.1016/j.celrep.2012.09.011, https://www.ncbi.nlm.nih.gov/pubmed/23063364, 23063364[pmid] 13

15. Jetley, S., Sapienza, M., Golodetz, S., Torr, P.H.S.: Straight to shapes: Real-time detection of encoded shapes. CoRR **abs/1611.07932** (2016), http://arxiv.org/abs/1611.07932 2
16. Keuper, M., Levinkov, E., Bonneel, N., Lavoué, G., Brox, T., Andres, B.: Efficient decomposition of image and mesh graphs by lifted multicuts. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1751–1759 (2015) 2
17. Kulikov, V., Lempitsky, V.: Instance segmentation of biological images using harmonic embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 1, 3, 8, 9
18. Lee, K., Lu, R., Luther, K., Seung, H.S.: Learning dense voxel embeddings for 3d neuron reconstruction (2019) 1
19. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 740–755. Springer International Publishing, Cham (2014) 9
20. Liu, Y., Yang, S., Li, B., Zhou, W., Xu, J., Li, H., Lu, Y.: Affinity derivation and graph merge for instance segmentation. CoRR **abs/1811.10870** (2018), http://arxiv.org/abs/1811.10870 1, 2
21. Ljosa, V., Sokolnicki, K.L., Carpenter, A.E.: Annotated high-throughput microscopy image sets for validation. Nature Methods **9**, 637 EP – (Jun 2012), https://doi.org/10.1038/nmeth.2083, correspondence 7, 8, 11
22. Long, F., Peng, H., Liu, X., Kim, S.K., Myers, E.: A 3d digital atlas of c. elegans and its application to single-cell analyses. Nature Methods **6**(9), 667–672 (2009). https://doi.org/10.1038/nmeth.1366, https://doi.org/10.1038/nmeth.1366 11
23. Maitin-Shepard, J.B., Jain, V., Januszewski, M., Li, P., Abbeel, P.: Combinatorial energy learning for image segmentation. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 1966–1974. Curran Associates, Inc. (2016), http://papers.nips.cc/paper/6595-combinatorial-energy-learning-for-image-segmentation.pdf 3
24. Nern, A., Pfeiffer, B.D., Rubin, G.M.: Optimized tools for multicolor stochastic labeling reveal diverse stereotyped cell arrangements in the fly visual system. Proceedings of the National Academy of Sciences **112**(22), E2967–E2976 (2015). https://doi.org/10.1073/pnas.1506763112, https://www.pnas.org/content/112/22/E2967 13
25. Novotny, D., Albanie, S., Larlus, D., Vedaldi, A.: Semi-convolutional operators for instance segmentation. In: The European Conference on Computer Vision (ECCV) (September 2018) 3, 8, 9
26. Ronneberger, O., P.Fischer, Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI). LNCS, vol. 9351, pp. 234–241. Springer (2015), http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a, (available on arXiv:1505.04597 [cs.CV]) 2, 5, 8
27. Schmidt, U., Weigert, M., Broaddus, C., Myers, G.: Cell detection with star-convex polygons. CoRR **abs/1806.03535** (2018) 3, 7, 11, 12, 13
28. Wählby, C., Kamentsky, L., Liu, Z.H., Riklin-Raviv, T., Conery, A.L., O'Rourke, E.J., Sokolnicki, K.L., Visvikis, O., Ljosa, V., Irazoqui, J.E., Golland, P.,

Ruvkun, G., Ausubel, F.M., Carpenter, A.E.: An image analysis toolbox for high-throughput c. elegans assays. Nature Methods **9**(7), 714–716 (2012). https://doi.org/10.1038/nmeth.1984, https://doi.org/10.1038/nmeth.1984 3, 9

29. Weigert, M., Schmidt, U., Haase, R., Sugawara, K., Myers, G.: Star-convex polyhedra for 3d object detection and segmentation in microscopy. arXiv:1908.03636 (2019) 3, 7, 12, 13
30. Wolf, S., Pape, C., Bailoni, A., Rahaman, N., Kreshuk, A., Köthe, U., Hamprecht, F.A.: The mutex watershed: Efficient, parameter-free image partitioning. In: ECCV (4). Lecture Notes in Computer Science, vol. 11208, pp. 571–587. Springer (2018) 1, 2, 5, 8, 10, 11, 12, 14
31. Wolf, S., Schott, L., Kothe, U., Hamprecht, F.: Learned watershed: End-to-end learning of seeded segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2011–2019 (2017) 1
32. Yurchenko, V., Lempitsky, V.S.: Parsing images of overlapping organisms with deep singling-out networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 4752–4760 (2017). https://doi.org/10.1109/CVPR.2017.505, https://doi.org/10.1109/CVPR.2017.505 2, 3, 8, 9