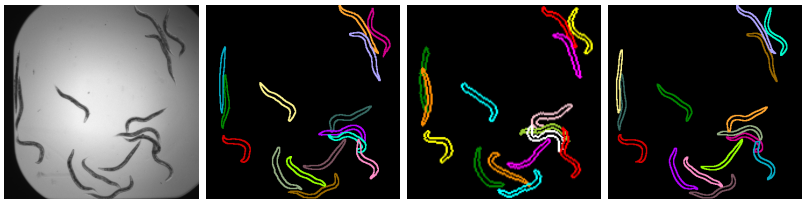
PatchPerPix for Instance Segmentation: Supplemental Material

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

Supplemental Figures for BBBC010 C. elegans worm disentanglement



(a) raw (b) ground truth (c) SON (d) PatchPerPix

Fig. 1: Qualitative comparison of PatchPerPix and Singling Out Networks [1] (SON) on a BBBC010 image. PatchPerPix (ppp+dec) is significantly more pixel-accurate as it does not rely on a dictionary of known shapes. In particular, it accurately separates a cluster of objects on the lower right. (SON image from [1])

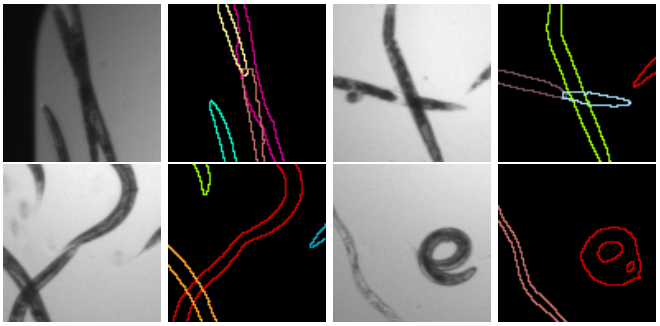


Fig. 2: Exemplary failure cases of PatchPerPix (ppp+dec) on BBBC010. (top) false split due to large overlap; false split due to missing signal. (bottom) false merge due to sequential layout of worms; inaccuracy due to strongly bent worm.

Supplemental Hyperparameter Studies on BBBC010

Table 1: Impact of patch size used in ppp+dec, assessed on BBBC010. All ppp+dec models have similar code size of ∼250, as well as a comparable number of parameters. For ppp+dec, AP at lower thresholds, as well as avAP, tends to increase with larger patch size. This is expected, as larger patches may bridge larger overlaps of instances. However, AP the highest threshold tends to decrease with larger patch size. This cannot be straightforwardly attributed to higher compression rate, as AP is robust to code size variation at fixed patch size (see Supp. Table 2). We hypothesize that it is due to shape variance increasing with increasing distance from the center pixel of a patch, causing larger patches to yield lower pixel accuracy at the fringes. Furthermore, ppp+dec considerably outperforms ppp at the same patch size (25x25) and same number of parameters. Improvement of ppp+dec over ppp is largest for AP at high thresholds. We hypothesize that this may be due to differences in training procedures, where in ppp+dec, only forground patches contribute to the loss, whereas in ppp, all patches contribute, thereby significantly shifting balance. Said hypotheses have to be verified by further experiments.

AP_{dsb}	patch size	$avAP_{[0.5:0.05:0.95]}$	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$
ppp	25x25	0.689	0.890	0.872	0.840	0.710	0.372
ppp+dec	25x25	0.720	0.895	0.877	0.857	0.763	**0.450**
ppp+dec	31x31	0.725	0.911	0.894	0.866	0.779	0.417
ppp+dec	41x41	0.727	**0.930**	0.905	0.879	0.792	0.386
ppp+dec	49x49	**0.736**	0.928	**0.914**	**0.898**	**0.802**	0.409

Table 2: Impact of code size used in ppp+dec, assessed on BBBC010. Patch size fixed at 41x41, number of parameters kept constant. ppp+dec achieves comparable results across a range of compression rates we assessed.

AP_{dsb}	code size	$avAP_{[0.5:0.05:0.95]}$	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$
ppp+dec	324	**0.737**	0.931	**0.919**	**0.897**	0.784	0.418
ppp+dec	252	0.727	0.930	0.905	0.879	0.792	0.386
ppp+dec	216	0.733	0.924	0.899	0.878	**0.794**	**0.425**
ppp+dec	180	0.730	0.924	0.906	0.888	0.786	0.413
ppp+dec	144	0.719	0.912	0.886	0.868	0.774	0.410
ppp+dec	108	0.734	**0.932**	0.914	0.893	0.776	0.409
ppp+dec	72	0.728	0.923	0.902	0.881	0.780	0.412

Supplemental Hyperparameter Study on DSB2018

Table 3: Impact of patch size and code size assessed on the 2d nuclei dataset dsb2018. In line with the respective study on BBBC010, results suggest that with a smaller patch size the network is able to better learn the exact instance shape (better at high IoU thresholds) yet with a larger patch size the detection performance improves (better at smaller IoU thresholds).

AP_{dsb}	patch size	code size	avAP [0.5:0.1:0.9]	$AP_{0.1}$	$AP_{0.2}$	$AP_{0.3}$	$AP_{0.4}$	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$
ppp	25	/	0.670	0.905	0.905	0.901	0.882	0.846	0.797	0.737	0.603	0.365
ppp+dec	25	252	**0.693**	0.919	0.919	0.915	0.898	0.868	**0.827**	0.755	**0.635**	**0.379**
ppp+dec	25	512	0.691	0.929	0.927	0.925	0.913	**0.874**	0.825	**0.763**	0.626	0.368
ppp+dec	41	252	0.682	0.924	0.921	0.919	0.898	0.871	0.824	0.744	0.613	0.359
ppp+dec	41	576	0.685	**0.934**	**0.934**	**0.931**	**0.916**	0.871	**0.827**	0.750	0.614	0.361

Supplemental Figure for Nuclei3d

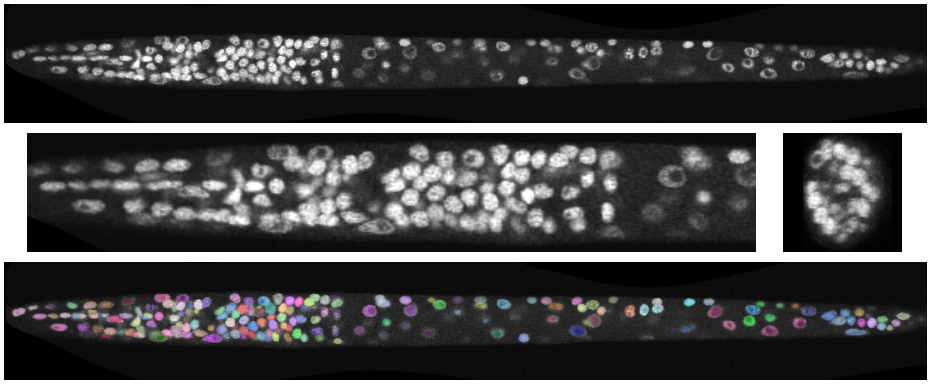


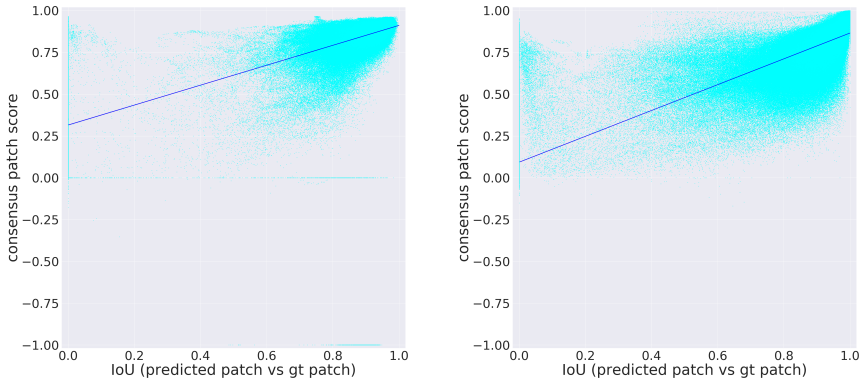
Fig. 3: Top: Exemplary xy-slice of a volume in the nuclei3d data set. Densely packed nuclei in the nervous system of the C. elegans L1 larva (towards the left) are particularly hard to separate. Center left: Close-up on said nervous system. Center right: Exemplary yz-slice of nervous system. Bottom: Respective PatchPerPix segmentation result.

Supplemental Study of Instance Assembly Run-times

Table 4: Average run-times for PatchPerPix instance assembly on the different datasets. We achieve convenient run-times with fair patch sizes on 2d data with sparse foreground (BBBC010 and dsb2018). Long run-time for ISBI2012 at fair patch size is due to the dense foreground of the data. Long run-time at small patch size for nuclei3d is due to the data being 3d, albeit with sparse foreground.

dataset	BBBC010	BBBC010	dsb2018	ISBI2012	nuclei3d
patch size	25x25	41x41	25x25	25x25	9x9x9
seconds per image	4	13	5	400	1300

Supplemental Analysis of Patch Scores



(a) BBBC010 (b) dsb2018

Fig. 4: We find significant correlation between the computed patch scores and the IoU of predicted patches vs. ground truth patches. Each plotted dot stems from a patch prediction at a pixel that is foreground either in the prediction or in the ground truth. Correlation values: 0.52 (Spearman's rho) and 0.37 (Kendall's tau) for BBBC010, and 0.8 (Spearman's rho) and 0.62 (Kendall's tau) for dsb2018.

Supplemental Figure for Instance Assembly

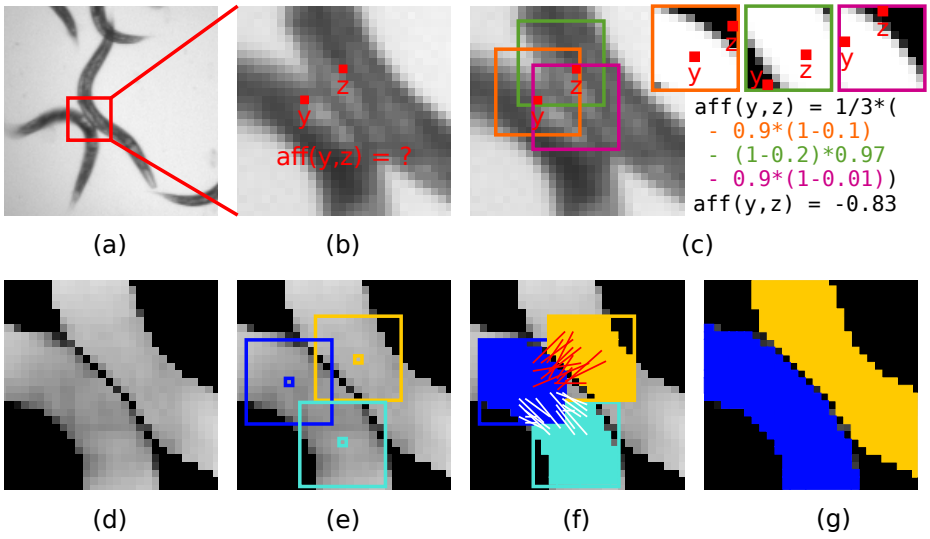


Fig. 5: Overview of our instance assembly pipeline: (a) Raw image detail showing two closely adjacent objects, (b) Zoom into ambiguous area. We use two pixels, y and z, delineated in red, to visualize how the consensus affinities aff(y,z) are computed. (c) Three exemplary patches, marked by different-colored squares, that each cover y and z. The actual value each contributes to aff(y,z) is stated next to the image (cf. Eq. 3). (d) Patch scores for all pixels, visualized as gray value image. Predictions closer to an ambiguous region between two objects, which agree less with the consensus affinities, receive lower patch scores (cf. Eq. 5). (e) Selection of high-scoring patches that contribute to cover the image foreground (for the sake of clarity only a subset is shown). They are depicted as an overlay, where the spatial extension of each selected patch is delineated by a colored box and the center pixel by a small square. (f) The crayoned areas show the foreground area covered by each patch. We compute patch affinities between overlapping patches (cf. Eq. 6), visualized by the lines (for the sake of clarity only a subset is shown). For the two patches connected by white lines the union of foregrounds agrees well with the consensus, they belong to the same object. For the two patches connected by red lines, the union of foregrounds does not agree well with the consensus, they belong to different objects. (g) The final instance segmentation.

References

1. 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 1