Volumetric Semantic Segmentation using Pyramid Context Features Supplemental Material

Jonathan T. Barron[1] Pablo Arbeláez[1] Soile V. E. Keränen[2]
Mark D. Biggin[2] David W. Knowles[2] Jitendra Malik[1]
[1]UC Berkeley [2]Lawrence Berkeley National Laboratory
{barron, arbelaez, malik}@eecs.berkeley.edu {svekeranen, mdbiggin, dwknowles}@lbl.gov

1. Complexity

Our "pyramid filtering" insight enables exact and extremely efficient per-voxel classification with minimal memory overhead. We will now demonstrate our improvement over existing techniques empirically and theoretically.

For an empirical demonstration of efficiency, consider the following two alternatives to pyramid filtering: 1) the "sliding window" approach: iterate through every voxel in the input volume, and for each voxel construct and classify a feature vector. 2) we could use the fact that our feature and classifier are both linear, and could therefore be reduced to a single linear operation which amounts to filtering the volume with an extremely large filter — one with a support as large as the input volume. Such filtering is intractably expensive in the spatial domain, but is much more efficient in the Fourier domain: we can compute the FFT of a volume and a filter, do a component-wise multiplication in Fourier space, and then invert the FFT. In the case in which we have multiple feature channels, this can be sped up by summing the filter-responses in Fourier space and inverting the FFT only once. We will refer to this as the "FFT" approach. The "FFT" approach can be sped up by assuming that the Fourier-domain filters have been precomputed, which reduces computational demands but dramatically increases memory overhead. We will refer to this precomputed Fourier technique as "FFT+caching". The sliding window approach is extremely common in object detection [5], and the FFT approach has also been explored previously [6]. And of course, there are many specialized ways to efficiently approximate these sliding-window type filtering operations [7, 8], but we will only consider general and exact techniques.

In Figure 1 we compare our pyramid filtering technique against the three previously-described baselines. We see that only our technique performs well in terms of speed and memory overhead when n is large — which is crucial, as $n \approx 256$ in our experiments. The sliding window technique has minimal memory overhead, but is intractably slow — often $200\times$ as slow as ours. The FFT techniques are faster, but still significantly slower than our technique by $5-20\times$, and have prohibitively large memory footprints ($6\times$ without caching, $160\times$ with caching) which caused our experiments to crash as n became large. Only our pyramid filtering technique can run efficiently (or, at all) on the volumetric data we are investigating.

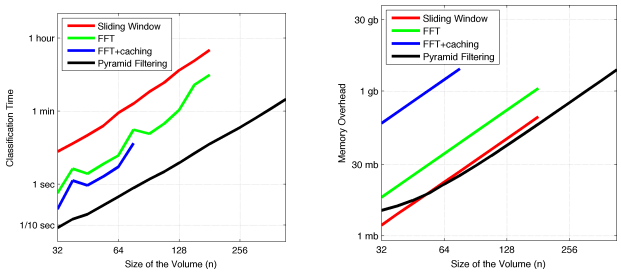


Figure 1. Profiles of different methods for densely evaluating a linear classifier on a feature vector for every voxel in a volume (where we have 50 channels for the volume, as is common in our experiments). Here we show the speed (time taken to classify the entire volume) and memory overhead (memory required to classify the volume, not including the feature channels of the volume themselves) for different classification techniques and different size volumes. Only our pyramid filtering technique is both fast and memory-efficient when n is large ($n \approx 256$ in our experiments). In fact, pyramid filtering is the only method which actually runs to completion when n is large – all other techniques run out of memory or never finish, which is why ours plots appear incomplete. Experiments were performed on a 2011 Macbook Pro.

Now that we have demonstrated the efficiency of pyramid filtering empirically, we will demonstrate it theoretically with regards to computational complexity. Consider a d-dimensional signal of size n, so in our experiments $d = 3$ and the signal is of size $(n \times n \times n)$. Let m be the width of the filter ($m = 3$ in our case), and f is the number of channels that the signal has been split into ($f \approx 50$ or 100 in our case). We assume the height of the pyramid being used is K, which is at most $\lfloor \log_2(n) \rfloor$. The computational

complexity of pyramid filtering is:

$$\frac{2^d\left(1-2^{-Kd}\right)}{2^d-1}fm^dn^d + Kn^d \tag{1}$$

In our case ($d = 3$, $K = 6$) this is approximately:

$$1.143fm^3n^3 + 6n^3 \tag{2}$$

Where the first term is the cost of convolving each level of the pyramid by a set of filters, and the second term is the cost of collapsing the pyramid (we omit the cost of constructing pyramids, as it is required by both our technique and the sliding window technique). We see that the cost is almost entirely dominated by that of convolving the largest scale of the pyramid (which is just the volume) by a small filter. Effectively, *we process every other scale for free*.

Now consider the computational complexity of the sliding window method for classifying every voxel in the volume:

$$2^dKfn^dm^d \tag{3}$$

where the 2^dK multiplier comes from having to do linear interpolation for each location on every level of the pyramid. In our case ($d = 3$, $K = 6$) this cost is:

$$48fn^3m^3 \tag{4}$$

The difference between our fast pyramid filtering technique and the traditional sliding window technique (ignoring the cost of collapsing a pyramid, which is negligible) is that our 1.143 multiplier has been replaced by 48. This tells us that we should expect at least a speedup of about $42\times$ relative to the sliding window technique.

Of course, our analysis ignores the fact that our fast technique is built upon convolutions with very small filters, which are generally fast and well-optimized. This is why we see a speedup of $200\times$ in practice, as opposed to the $42\times$ improvement our analysis predicts.

2. Feature Learning

One of the feature channels used in our model is a "adaptive" set of features based on raw pyramid-context features of the input volume. To learn these features, we use the simple feature-learning technique of [4] to learn filters, which is effectively whitening and k-means. First, we will extract a set of pyramid-context features $\{x_i\}$ from our raw training volumes. To whiten, we compute the sample mean and covariance matrix of $\{x_i\}$:

$$\mu = \frac{1}{|x|}\sum_i x_i \tag{5}$$

$$\Sigma = \frac{1}{|x|}\sum_i (x_i-\mu)(x_i-\mu)^{\mathrm{T}} \tag{6}$$

The whitening matrix is:

$$W = V\,\mathrm{diag}(1/\sqrt{\lambda+\epsilon})V^{\mathrm{T}} \tag{7}$$

Where V is a matrix of eigenvectors of Σ, λ is a vector of eigenvalues, and ϵ is a small constant to deal with eigenvalues that are close to 0, (which we set to 0.01, and which feature-learning is surprisingly sensitive to). With this transformation we can whiten our data:

$$w_i = W(x_i - \mu) \tag{8}$$

We will then learn a dictionary from our whitened data, by solving the following optimization problem.

$$\begin{aligned} \underset{d,c}{\text{minimize}} \quad & \sum_i \|dc_i - w_i\|_2^2 \\ \text{subject to} \quad & \|d_j\|_2^2 = 1, \forall_j \\ & \|c_i\|_0 \le 1, \forall_i \end{aligned} \tag{9}$$

This is the "OMP-1" or "gain-shape vector quantization" algorithm used in [4]. It closely resembles k-means, except that similarly to cluster j is measured by the inner product $\langle w_i, d_j\rangle$, instead of Euclidian distance $\|w_i - d_j\|_2^2$. If each w_i is unit length (which they are not, though whitening produces vectors that have similar magnitudes) then this algorithm can be viewed as minimizing the angle between each datapoint and its cluster. We solve Equation 9 using a greedy k-means-like coordinate descent algorithm: Each datapoint w_i is assigned greedily to whichever dictionary element d_j best minimizes the residual reconstruction error $\|d_jc_i - w_i\|_2^2$, which gives us c_i, the "cluster assignment" of datapoint w_i. Once we have all c_i, we update each d_j to be the data, weighted by the strength of each cluster assignment: $d_j = \sum_i w_ic_i(j)$. We then renormalize each d_j to be unit-length, and repeat the procedure.

Once we have computed all d_j, we invert the whitening transformation to get unwhitened filters $f_j = Wd_j$. Because of the centering in whitening, we must also compute a set of biases: $b_j = -\mu^{\mathrm{T}}Wd_j$. With this, we can compute our feature channels $\{F\}$ as follows:

$$F^{(j)} = \max(0, (V \otimes f_j) + b_j) \tag{10}$$

Where $\otimes$ is our "pyramid-filtering" procedure, as described in the paper. The $\max(0,\cdot)$ term is a "soft thresholding" which serves to introduce a non-linearity, and which was found to be the most effective encoding technique in [4].

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 2002.

[2] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. *CVPR*, 2005.

[3] A. C. Berg and J. Malik. Geometric blur for template matching. *CVPR*, 2001.

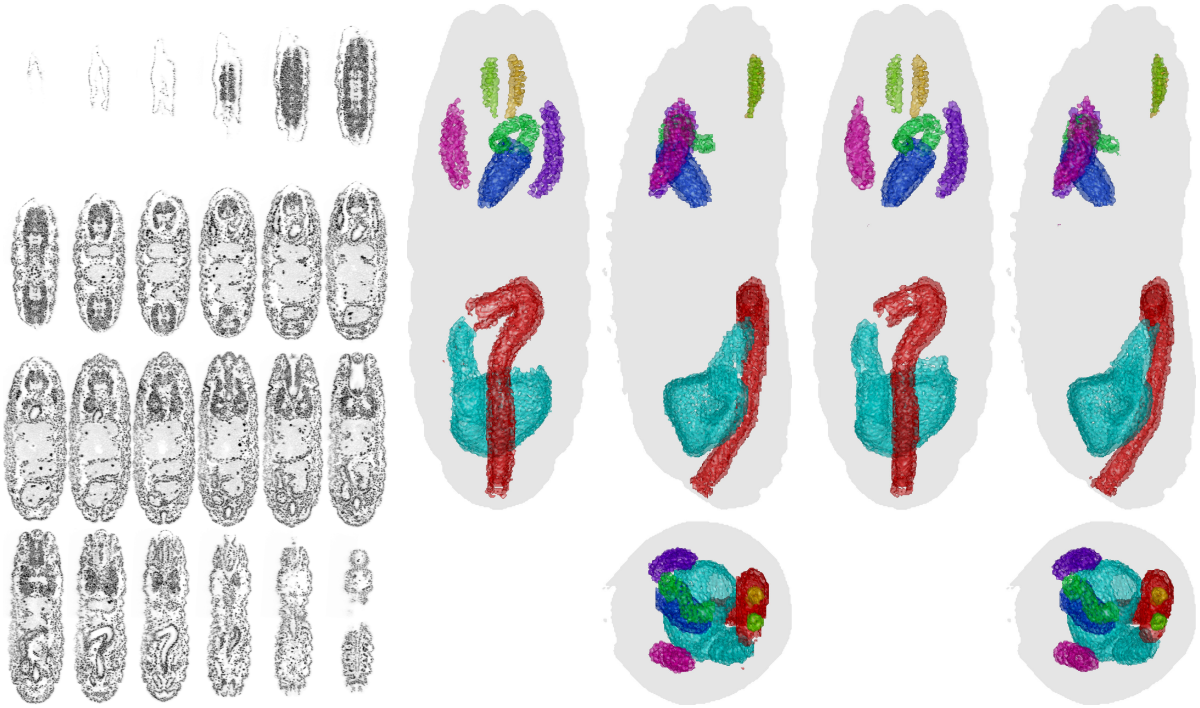
[4] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. *ICML*, 2011.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *ICCV*, 2005.

[6] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. *ECCV*, 2012.

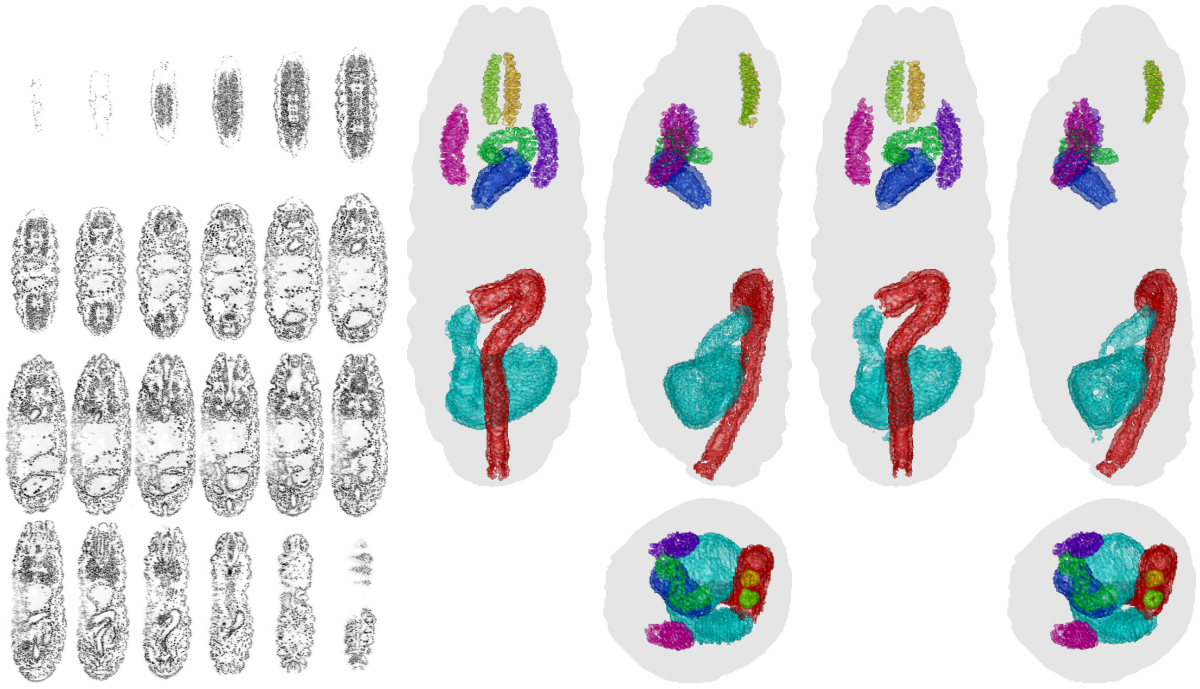
[7] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. *CVPR*, 2010.

[8] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.



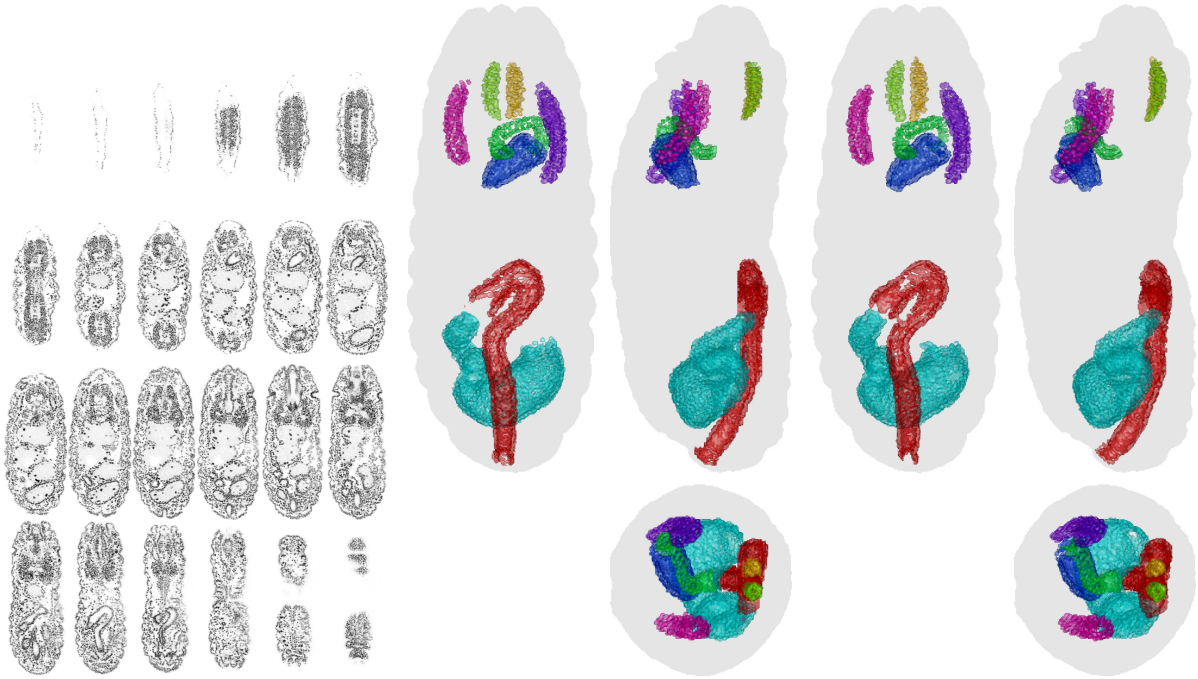
(a) Input Signal (b) Ground Truth (c) Our Prediction

Figure 2. In Figure 2(a) we have slices of a fluorescent volume of a late-stage *Drosophila* embreyo. In Figure 2(b) we have annotations of 8 biologically-relevant tissues from a biologist. Given the volume in Figure 2(a) we can produce a per-voxel prediction of each tissue from a new (test-set) volume in a matter of minutes, as shown in 2(c).



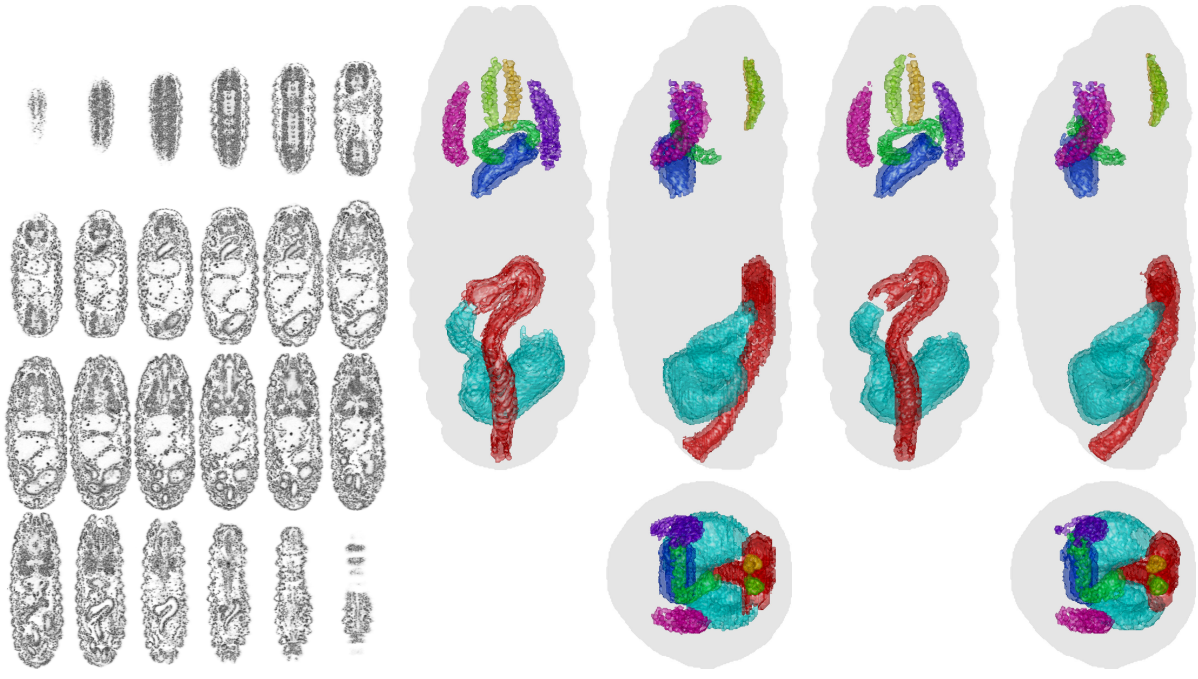
(a) Input Signal (b) Ground Truth (c) Our Prediction

Figure 3. Another test-set volume, shown in the same fashion as Figure 2.



(a) Input Signal (b) Ground Truth (c) Our Prediction

Figure 4. Another test-set volume, shown in the same fashion as Figure 2.



(a) Input Signal (b) Ground Truth (c) Our Prediction

Figure 5. Another test-set volume, shown in the same fashion as Figure 2.

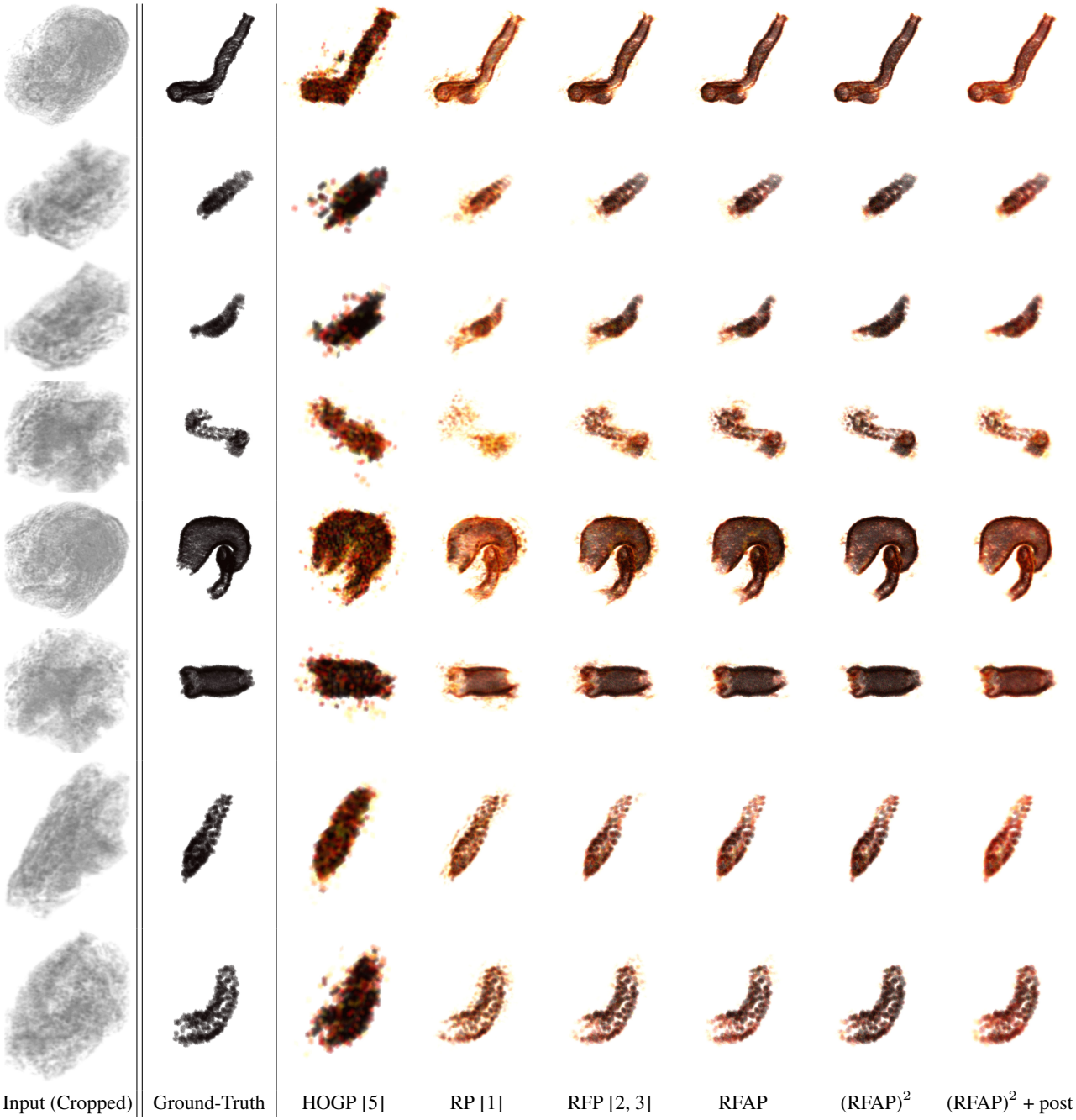


Figure 6. Some visualizations of the output of our model, and other models, on a test-set volume. In the first column we have the input volume (cropped near the tissue of interest), and in the second we have the ground-truth annotation of a tissue. The other columns are the output of various models, the first being an improved HOG baseline, the last being our complete model, and the others being notable ablations of our model (some of which resemble optimized and improved versions of other techniques). Here we show the 8 tissues for a single test-set volume.

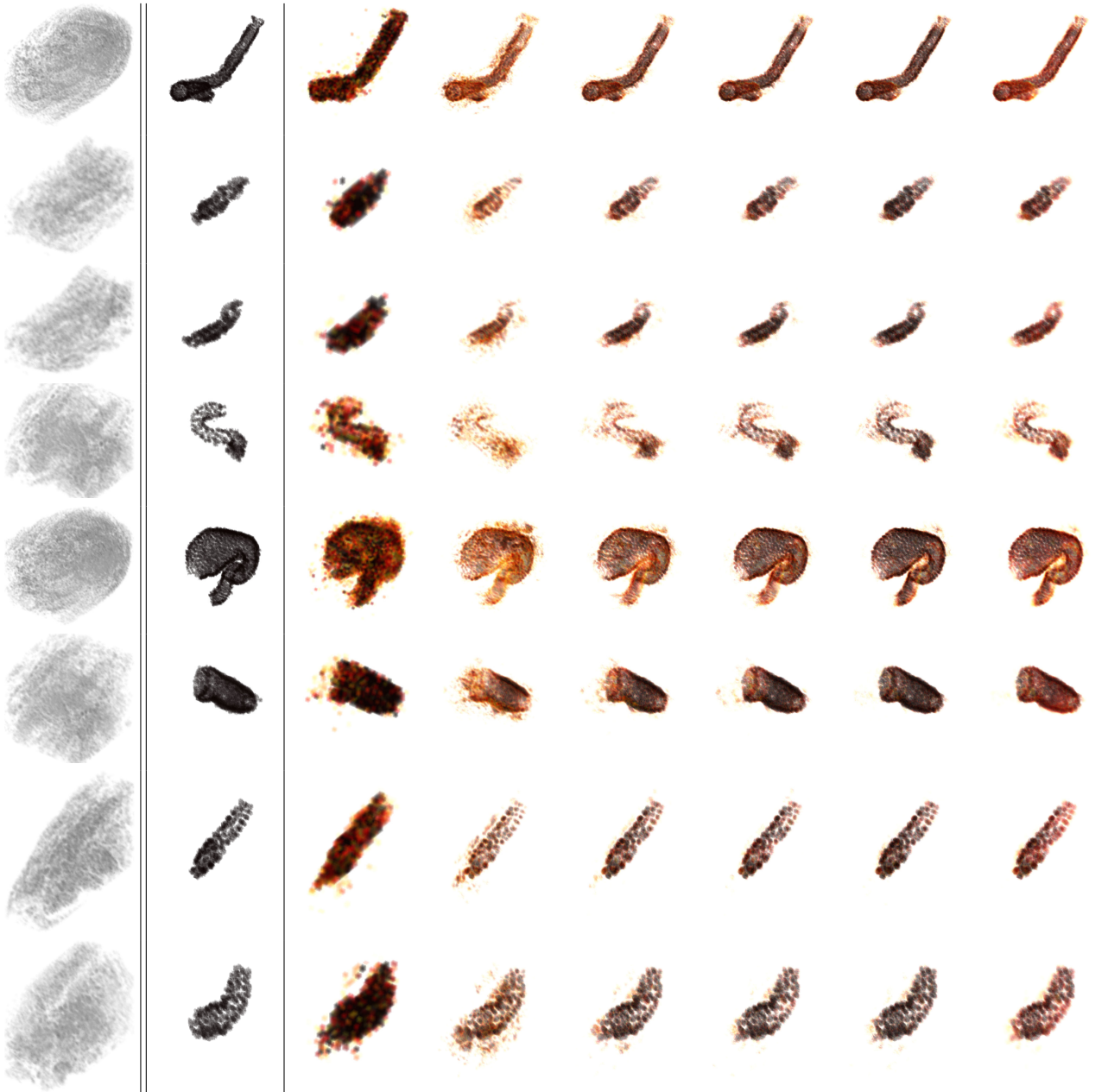


Figure 7. This figure has the same structure as Figure 6, but for a different test-set volume.

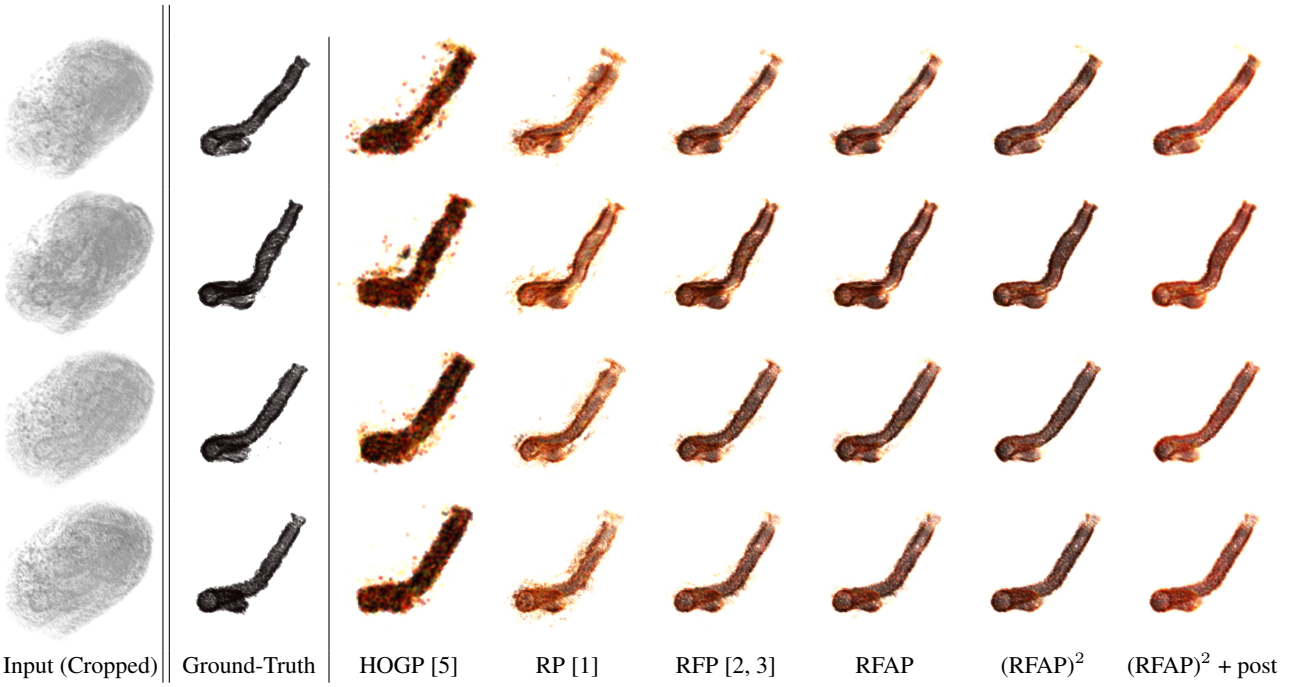


Figure 8. This figure has the same structure as Figure 6, but here we have just one tissue(tissue 1), and we show several test-set volumes (one per row) for just that tissue. We see that the in-class variation of these tissues is substantial.

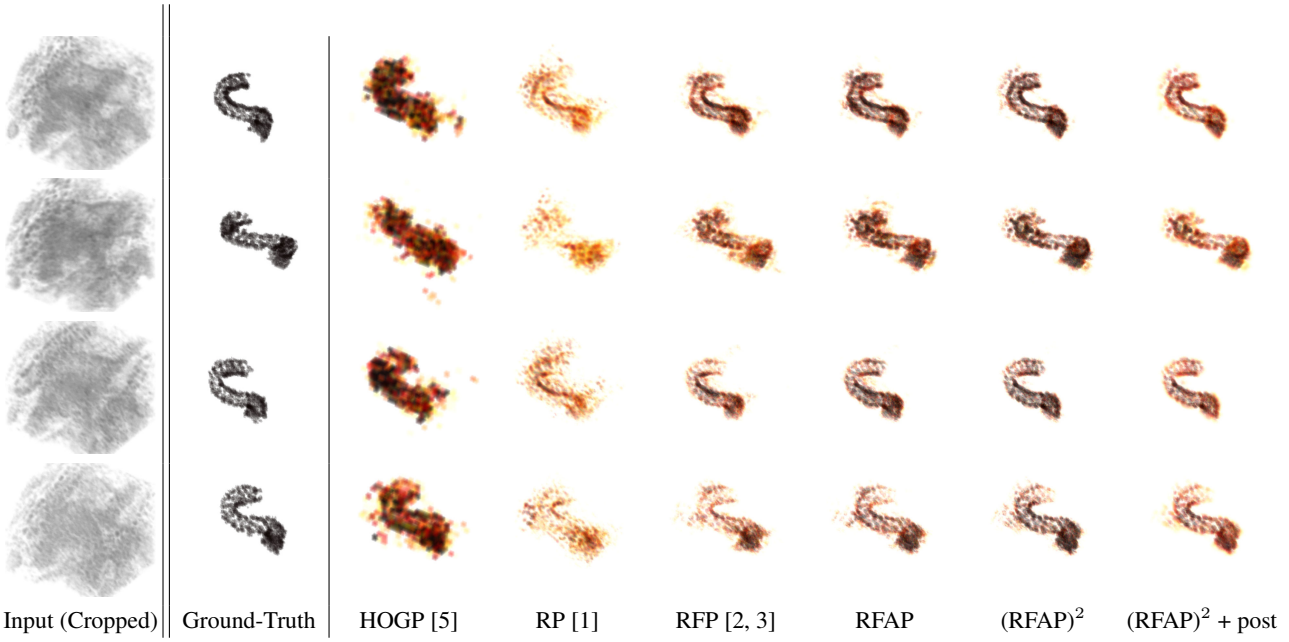


Figure 9. This figure has the same structure as Figure 8, but for tissue 4.

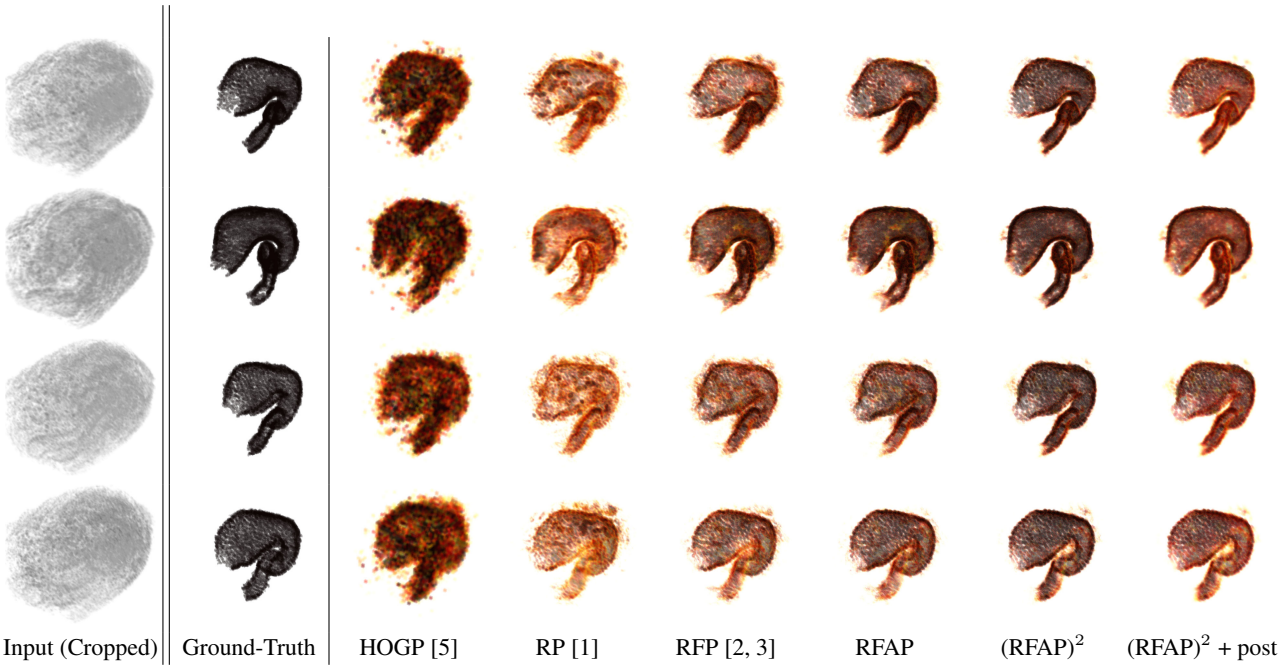


Figure 10. This figure has the same structure as Figure 8, but for tissue 5.

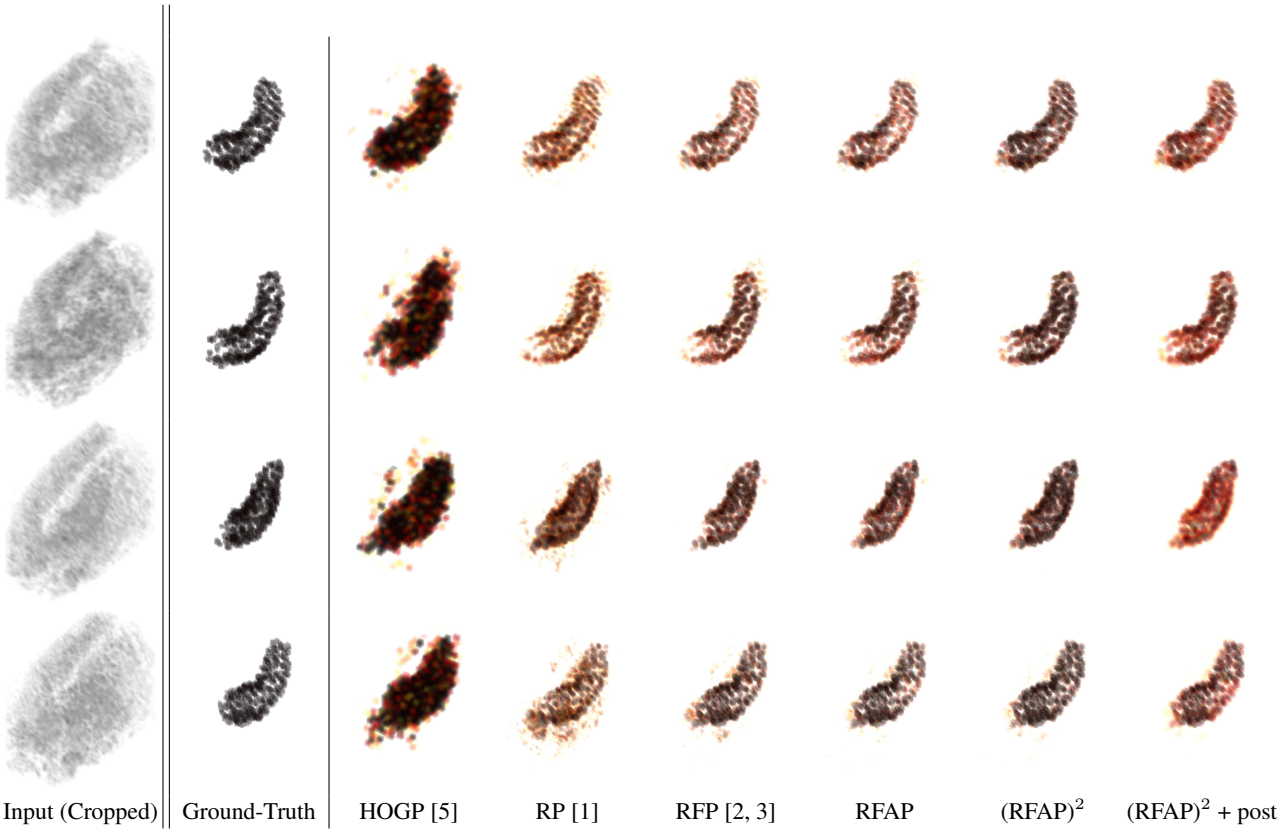


Figure 11. This figure has the same structure as Figure 8, but for tissue 8.

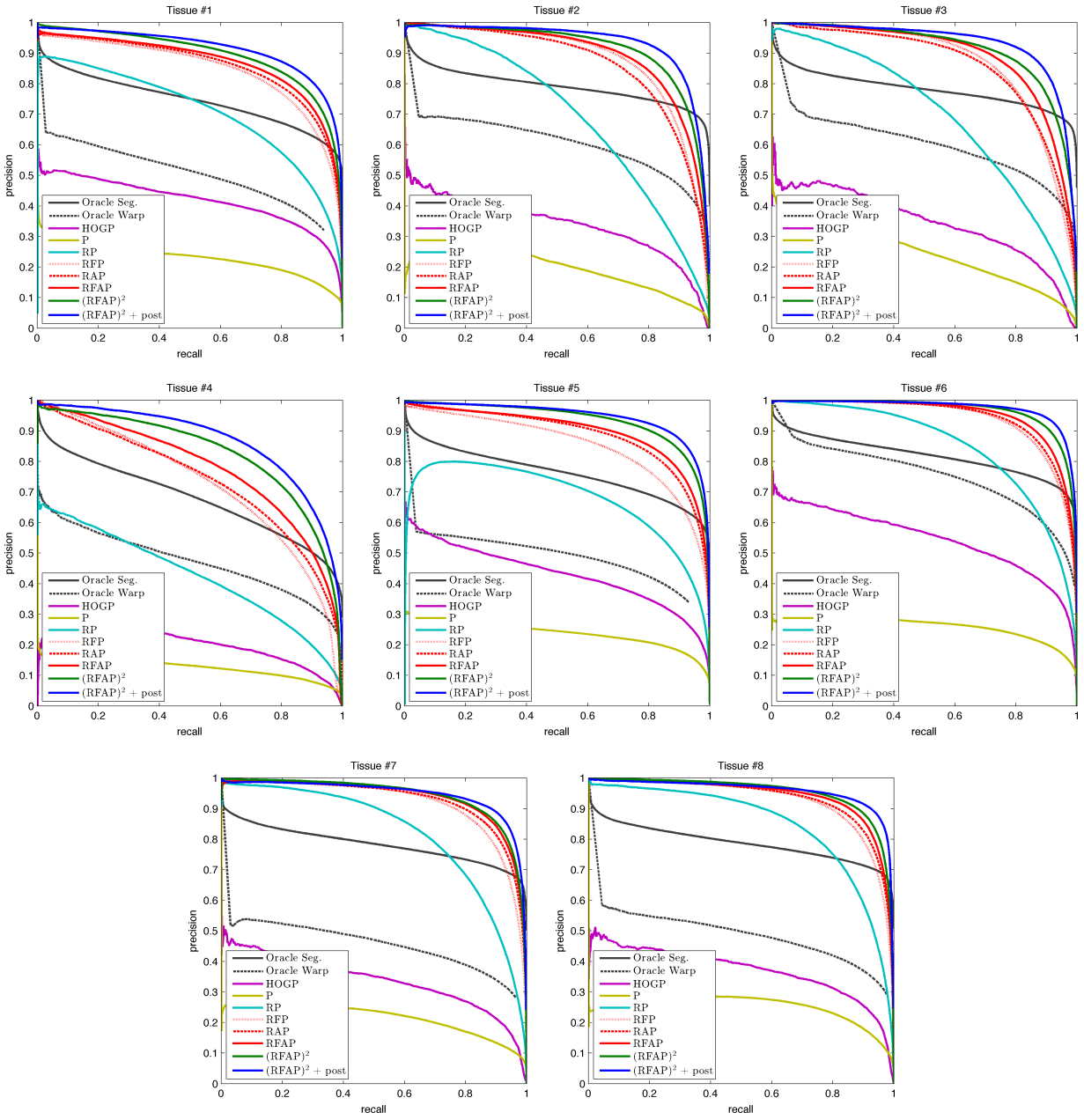
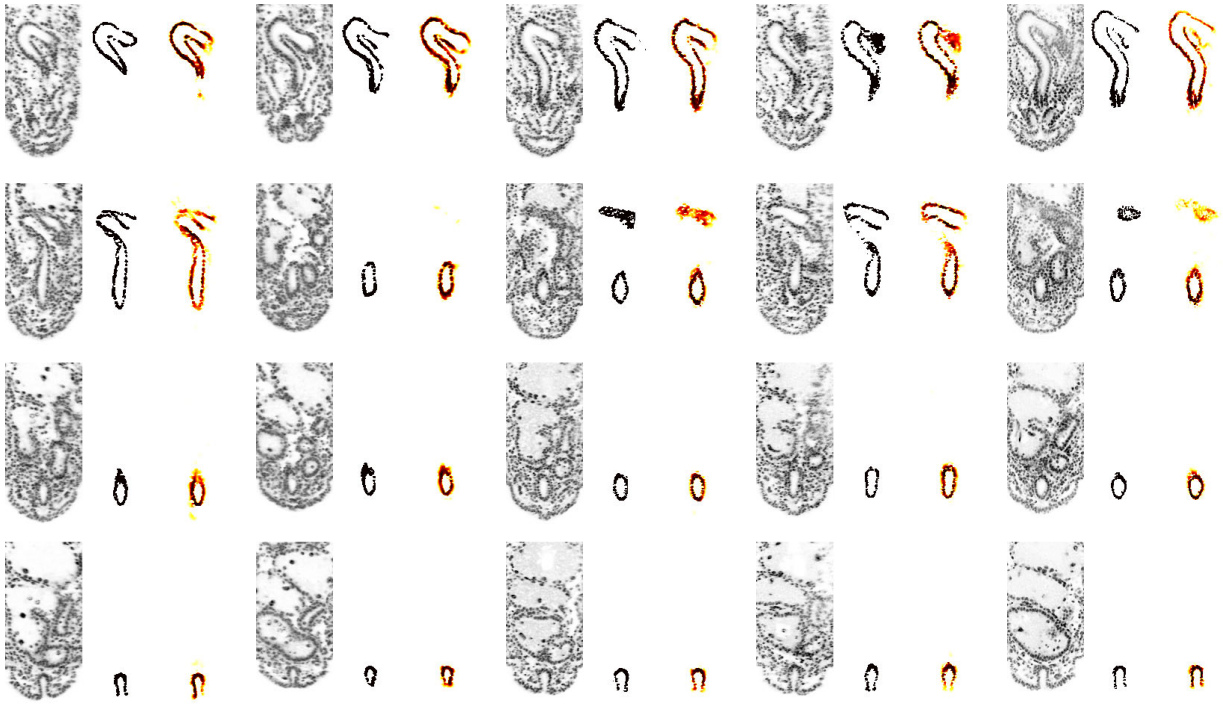
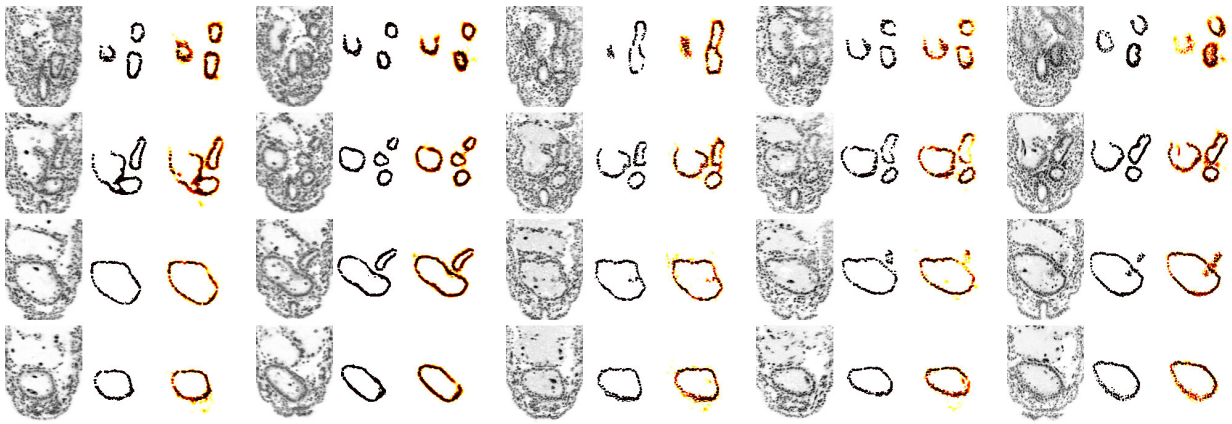


Figure 12. Precision/recall curves for different models on our entire test set, with one plot for each specific tissue.



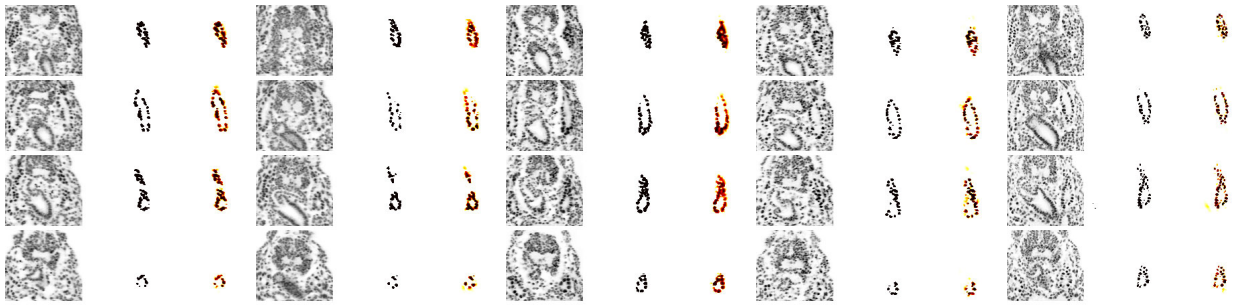
(a) Test Volume 1 (b) Test Volume 2 (c) Test Volume 3 (d) Test Volume 4 (e) Test Volume 5

Figure 13. Because of the difficulties in visualizing volumes, here we show slices of our volume, for one tissue (tissue 1), across 5 test-set volumes. The left column is a slice of the input volume, the middle column is the ground-truth annotation, and the right column is our prediction. This demonstrates the difficulty of the task that a biologists faces when viewing or annotating a volume — identifying which pixels in a slice belong to a tissue is challenging, especially when looking at only a single cross-section of the volume (of course, our model looks at the entire volume at once when labeling voxels). We see that our model does a very good job of predicting tissues, even at a per-voxel degree of accuracy. We also see the significant intra-class variation for a given tissue — these slices are all drawn from the same positions in our "canonically" aligned volumes, but the the same slice of the same tissue varies dramatically from volume to volume.



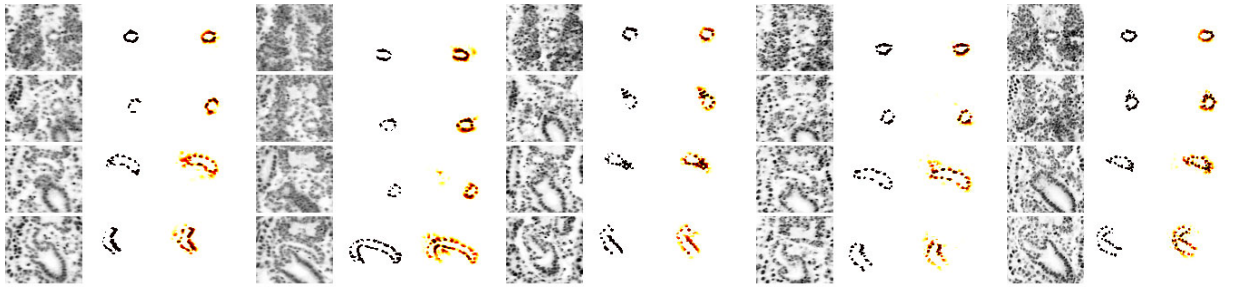
(a) Test Volume 1 (b) Test Volume 2 (c) Test Volume 3 (d) Test Volume 4 (e) Test Volume 5

Figure 14. This figure has the same structure as Figure 13, but for tissue 5.



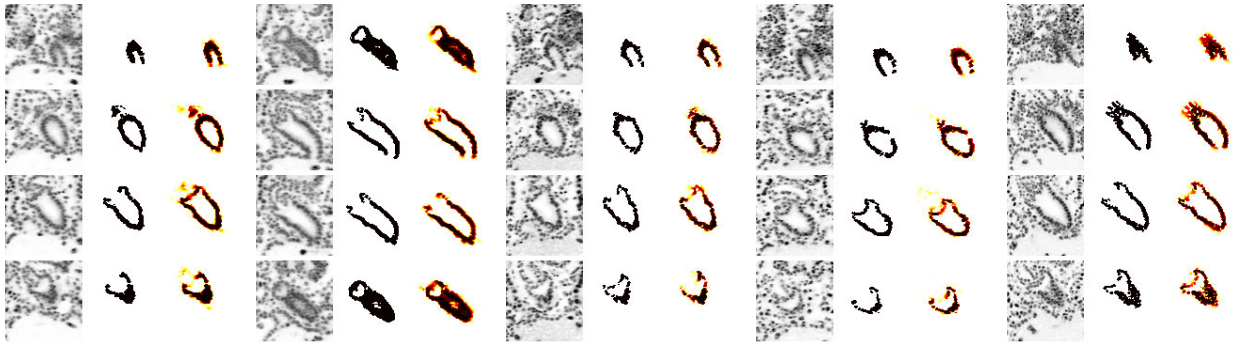
(a) Test Volume 1 (b) Test Volume 2 (c) Test Volume 3 (d) Test Volume 4 (e) Test Volume 5

Figure 15. This figure has the same structure as Figure 13, but for tissue 8.



(a) Test Volume 1 (b) Test Volume 2 (c) Test Volume 3 (d) Test Volume 4 (e) Test Volume 5

Figure 16. This figure has the same structure as Figure 13, but for tissue 4.



(a) Test Volume 1 (b) Test Volume 2 (c) Test Volume 3 (d) Test Volume 4 (e) Test Volume 5

Figure 17. This figure has the same structure as Figure 13, but for tissue 6.