# Volumetric Semantic Segmentation using Pyramid Context Features

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

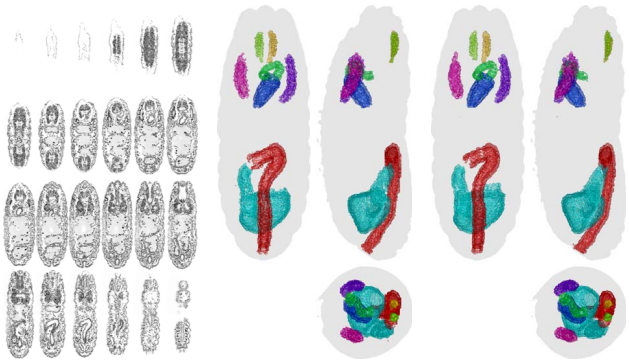
## Abstract

*We present an algorithm for the per-voxel semantic segmentation of a three-dimensional volume. At the core of our algorithm is a novel "pyramid context" feature, a descriptive representation designed such that exact per-voxel linear classification can be made extremely efficient. This feature not only allows for efficient semantic segmentation but enables other aspects of our algorithm, such as novel learned features and a stacked architecture that can reason about self-consistency. We demonstrate our technique on 3D fluorescence microscopy data of Drosophila embryos for which we are able to produce extremely accurate semantic segmentations in a matter of minutes, and for which other algorithms fail due to the size and high-dimensionality of the data, or due to the difficulty of the task.*

## 1. Introduction

Consider Figure 1(a), which shows slices from a volumetric image of a fruit fly embryo in its late stages of development, acquired with 3D fluorescence microscopy. Such data is a cornucopia of knowledge for biologists, as it provides direct access to the internal morphology of a widely studied model organism at an unprecedented level of detail. Traditionally, such information is encoded in a morphological atlas (for *Drosophila*, see [7]), which is painfully constructed by physically slicing embryos and manually annotating each tissue. However, the recent availability of high-resolution volumetric images from multiple modalities has spurred a great interest in the scientific community for the creation of "virtual atlases" [15, 16, 20], typically relying on the semantics provided by interactive segmentation or gene expression patterns. From a computer vision perspective, the problem at hand is that of volumetric semantic segmentation, in which we must predict a tissue label for each voxel in a volume. In this paper, we present an extremely accurate and efficient algorithm for volumetric semantic segmentation, based on a novel feature type called the "pyramid context". Figure 1(b) presents ground-truth annotations manually collected by an expert for 8 key morphological structures, and Figure 1(c) shows the results of our approach on this test-set volume.

The state-of-the-art in semantic segmentation on 2D images is represented by the leading techniques on the PASCAL VOC challenge [14]. The best performing methods, *e.g.* [2, 8, 9] operate by classifying object candidates obtained by expensive bottom-up grouping. They use representations tailored to capture the appearance of common objects (*e.g.* colorSIFT [24]), and the output of pre-trained object detectors [2], combined with non-linear classifiers [2, 9] or, alternatively, high-dimensional second order features [8]. A second family of approaches, based on CRFs, *e.g.* [6], extends such pixel-wise classifiers by modeling also pairwise dependencies, co-occurrence statistics, or higher-order potentials. All such techniques, which build



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

Figure 1. We will address the task of taking a volumetric scan of an object (in our case, a late-stage *Drosophila* embryo, see 1(a) for a visualization of some of the constituent "slices" of the volume, where the upper left slice is the top of the embryo and the bottom right slice is the bottom) and producing a per-voxel semantic segmentation of that volume. Given training annotations of 8 tissues or organs from a biologist, such as in 1(b), we can produce a per-voxel prediction of each tissue from a new (test-set) volume in a matter of minutes, as shown in 1(c). Many more such figures can be found in the supplementary material.

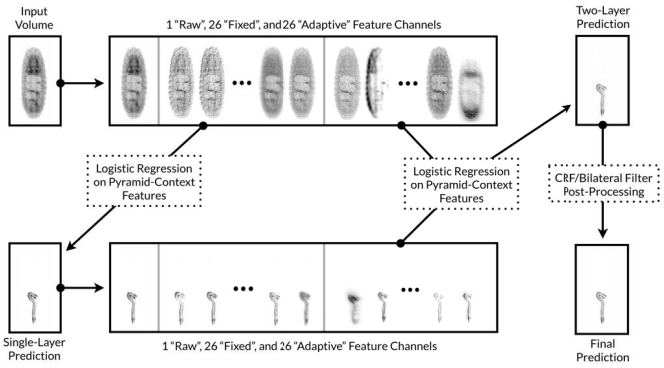



Figure 2. An overview of our pipeline. Our classification architecture consists of two layers. Our first layer takes as input 4 feature types computed from the input volume (top row, position features are not shown) to produce a per-voxel prediction. This output is fed to a second layer, which computes the same types of features from that per-voxel prediction, and uses the first-layer features with the new second-layer features (bottom row) to produce a new prediction. The output of the two-layer model is then smoothed using a joint-bilateral filter. See Section 3.1 for an explanation of the different feature channels shown here.

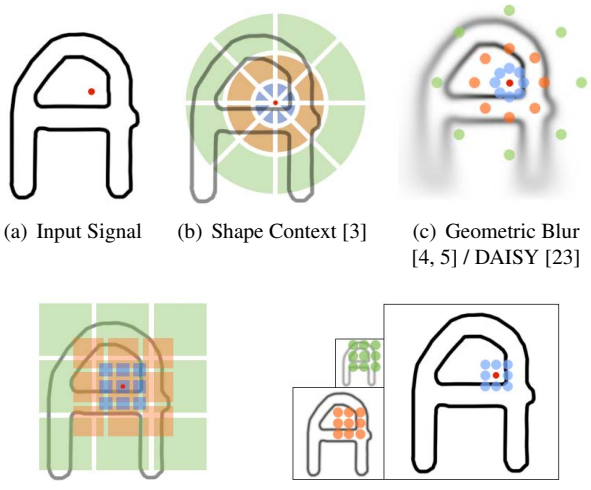
upon decades of computer vision research on 2D natural images, are simply intractable and inapplicable in the *terra incognita* of volumetric semantic segmentation: sophisticated 2D segmentation techniques break down when faced with 15 million voxels, and simple approaches like watersheds produce segments which are too coarse for the accurate per-voxel labeling of extremely fine-scale biological structures. Traditional sliding-window detection techniques [12] are intractably expensive to densely evaluate at every window in a 15 megavoxel volume, and generally reason only about local appearance, not large-scale context. The handful of volumetric segmentation techniques which do exist are restricted to the specific task of connectomics with Electron Microscopy [1, 25, 26].

Because existing techniques are insufficient, we must construct a novel semantic segmentation algorithm. We will address the problem as one of evaluating a classifier at every voxel in a volume. Our features must be descriptive enough to differentiate between fine-scale structures while spatially large enough to incorporate coarse-scale contextual information, and per-voxel classification of our features must be efficient. To address these issues we introduce the "pyramid context" feature, which can be thought of as a variant of retina-like log-polar features such as the shape context [3]. A key property of this feature is that by design, the dense evaluation of a linear classifier on pyramid context features is extremely efficient. To create a semantic-segmentation algorithm, we will construct these pyramid context features using oriented edge information (as in HOG [12] or SIFT [21]) and also learned "codebook" like features (as in a bag-of-words models [18]). We can then stack these pyramid context layers into a multilayer architecture which allows our model to reason about context and self-consistency. A visualization of our semantic-segmentation pipeline can be seen in Figure 2.

Our results are extremely accurate, with per-voxel APs in the range of 0.86-0.98 — accurate enough that our test-set predictions are often indistinguishable from our ground-truth by trained biologists. Our model is fast — evaluation of a volume takes a matter of minutes, while the time taken by a biologist to fully annotate an embryo is often on the order of hours, and the time taken by existing computer vision techniques is on the order of days. And our model is exact — we gain efficiency not through approximations or heuristics, but by designing our features such that exact efficient classification is possible.

## 2. The Pyramid Context Feature

At the core of our algorithm is our novel "pyramid context" feature. The pyramid context is similar to the shape context feature [3], geometric blur [4, 5], or DAISY features [23] — all serve to pool information around a location in a log-polar arrangement (Figure 3). The key insight behind our pyramid context feature is that there exist two equivalent "views" of the feature: it can be viewed as a Haar-like



(a) Input Signal (b) Shape Context [3] (c) Geometric Blur [4, 5] / DAISY [23]

(d) Pyramid Context (e) Pyramid Context

Figure 3. Given an input signal and a location (3(a)) we can pool local information in a retina-like fashion to construct a feature, such as shape context (3(b)) or geometric blur / DAISY (3(c)). We present a novel feature type, the "pyramid context" (3(d)) which can be thought of as a pyramid/Haar-like generalization of past pooling features. The key insight of this paper is that this feature can be re-expressed as efficient local operations on a Gaussian pyramid of a signal (3(e)), which allows us to *extremely efficiently* evaluate a linear classifier on pyramid context features at *every pixel* in the image using simple pyramid operations and convolutions with very small kernels.

pooling of signals at different scales (Figure 3(d)) or as a series of interpolations into a Gaussian pyramid of a signal (Figure 3(e)). Because of this, we can evaluate a linear classifier on top of pyramid context features at every voxel in a volume extremely efficiently, using simple pyramid operations and convolutions with very small kernels. In this section we will formalize our feature, and present an efficient per-voxel classification algorithm which is orders of magnitude faster than existing alternatives.

Let $V$ be a volume, and let us define $c(V, x, y, z)$, which computes a feature vector from $V$ at location $(x, y, z)$:

$$\begin{aligned} c(V,x,y,z) \quad = [ \quad & V(x+1, y+1, z+1); \\ & V(x, y+1, z+1); \\ & \ldots \\ & V(x, y-1, z-1); \\ & V(x-1, y-1, z-1)] \end{aligned} \tag{1}$$

Where $V(x, y, z)$ is the linearly-interpolated value of volume $V$ at location $(x, y, z)$. $c(\cdot)$ simply vectorizes a $3\times3\times3$ region of a volume into a vector. Note that the offsets are ordered such that $\langle \mathbf{w}, c(V, x, y, z)\rangle = (V * \mathbf{w})_{x,y,z}$ [1]. This means that a linear classifier on top of these features can be reformulated as a convolution of the volume.

Now let $P(V)$ be a $K$-level Gaussian pyramid of $V$, such that $P_k(V)$ is the $k$-th level of the pyramid ($P_1(V) = V$). A pyramid context feature is the concatenation of our simple "context" features at every scale of the pyramid:

$$\begin{aligned} C(V,x,y,z) \quad = [ \quad & c\left(V, x, y, z\right); \\ & c\left(P_2(V), \tfrac{x}{2}, \tfrac{y}{2}, \tfrac{z}{2}\right); \\ & \ldots \\ & c\left(P_K(V), \tfrac{x}{2^{K-1}}, \tfrac{y}{2^{K-1}}, \tfrac{z}{2^{K-1}}\right)] \end{aligned} \tag{2}$$

Where $K = 6$ in our experiments.

Consider a linear classifier for pyramid context features. To classify every voxel in a volume, we must compute $\langle \mathbf{w}, C(V, x, y, z)\rangle$ for all $(x, y, z)$. Doing this naively is extremely inefficient: the volume is extremely large (15 million voxels), and the corresponding features for each voxel are hard to compute: each requires hundreds of trilinear interpolation operations into a pyramid.

To make this problem tractable, we leverage the fact that every operation in this architecture is linear, and therefore associative. Instead of calculating $\langle \mathbf{w}, C(V, x, y, z)\rangle$ for all $(x, y, z)$, we convolve each level of $P(V)$ with $\mathbf{w}_k$, the subset of $\mathbf{w}$ that corresponds to level $k$, reshaped into a $3\times3\times3$ filter. Once we have a filtered Gaussian pyramid, we collapse the pyramid by upsampling each scale to the size of the volume, and summing the upsampled scales. We will refer to this process (computing $P(V)$, filtering each $P_k(V)$ with $\mathbf{w}_k$, and collapsing the filtered $P(V)$ to a volume) as $V \otimes \mathbf{w}$, or as "pyramid filtering" $V$ with $\mathbf{w}$.

[1] In a slight abuse of notation, $\mathbf{w}$ will simultaneously be referred to as a vector and as a $3 \times 3 \times 3$ filter

Instead of learning classifiers directly on the input volume $V$ we will produce a set of "feature channels" $\{F\}$ from $V$, pyramid filter each channel with its own set of weights $\mathbf{w}^{(j)}$, and sum those together to produce our per-voxel prediction: $G = \sum_j F^{(j)} \otimes \mathbf{w}^{(j)}$. This can be made much faster by noticing that the pyramid collapse at the end of each pyramid filtering is linear, and so we can sum up the filtered pyramids and then collapse the summed pyramid only once. Formally, pseudocode for our efficient per-voxel classification is:

$G \leftarrow 0$
**for** $k = [1 : K]$ **do**
  $G_k \leftarrow 0$
  **for** $j = [1 : |\{F\}|]$ **do**
    $G_k \leftarrow G_k + P_k(F^{(j)}) * \mathbf{w}_k^{(j)}$
  $G \leftarrow G + \texttt{upsample}(G_k)$
**return** $G$

See the supplementary material for a demonstration of the improvement in efficiency yielded by using "pyramid filtering" instead of pre-existing techniques, such as sliding-window [12] or FFT-based filtering [13]. Empirically our technique is $200\times$ faster than sliding window while having nearly as small a memory footprint, and is $5\times$ to $20\times$ faster than FFT-based techniques while requiring $1/6$th or $1/160$th the memory. In short, only pyramid filtering can run efficiently (or, at all) on the volumetric data we are investigating — naive alternatives either take over 1.5 hours or require over a hundred gigs of memory, while our technique takes less than 30 seconds and requires less than 1 gigabyte of memory. Analytically, we show through complexity analysis that pyramid filtering should be $42\times$ as fast as sliding-window, though we see a much greater improvement in practice because small convolutions are generally fast for non-algorithmic reasons (memory locality, optimized code, etc).

## 3. Semantic Segmentation Algorithm

We will now build upon our novel feature descriptor and its corresponding efficient classification technique to construct a volumetric semantic-segmentation algorithm, as shown in Figure 2. In Section 3.1 we will present three kinds of feature channels for use as input to our model, some of which are themselves built upon pyramid context features. In Section 3.2 we present an additional feature type based on the absolute position of each voxel. In Section 3.3 we will show how to use the output of a single-layer classification model built on the features in Sections 3.1 and 3.2 to build a two-layer model which uses contextual information, again by exploiting our pyramid context features. In Section 3.4 we present a post-processing step based on joint-bilateral filtering.

### 3.1. Feature Channels

The simplest feature-channel which we can use is the raw input volume, which we will refer to our "raw" feature channel. We augment this channel with two kinds of feature channels computed from the raw input volume: a "fixed" type based on simple first and second derivatives of the input volume (similar to HOG [12] or SIFT [21]) and a novel "adaptive" type learned from pyramid context features on top of the raw volume.

To compute our "fixed" features we take our volume $V$, compute a Gaussian pyramid $P(V)$, convolve each level by a set of filters, half-wave rectify the output [22], and concatenate the channels together[2]. The filters we use are just 3-tap oriented gradient filters in all directions (12 in all), and the 3D discrete Laplace operator. For each filter $f$, we convolve each pyramid level $P_k(V)$ with that filter, and produce the following two channels:

$$\max(0, P_k(V) * f), \quad \max(0, -(P_k(V) * f)) \tag{3}$$

giving us a total of 26 channels. Examples of our "fixed" channels can be seen in Figure 2.

Though these simple filter responses are powerful, they are limited. They describe coarse first or second order variation of the volume, but do not, for example, describe local context, or the distribution of the signal at multiple scales at the same location. It is difficult to use one's intuition to hand-design appropriate features, especially in unexplored domains such as our volumetric fluorescence data, so we will use semi-supervised feature learning to learn our second set of "adaptive" feature channels.

Traditional feature-learning techniques usually involve learning a set of filters from image patches [11, 19]. On our data, these techniques fail for the same reasons that naive classification fails: the sheer size and high-dimensionality of our data makes basic techniques intractable. Filtering volumes with the medium-sized filters commonly used in feature learning experiments ($9 \times 9$, $14 \times 14$, etc) is intractable, and such filters have too small a spatial support to provide useful information regarding context or morphology. We will therefore use our pyramid context features as a *substrate* for feature learning: we will extract pyramid context features from the raw volume, learn a set of filters for those features, and then pyramid filter the volume according to those learned filters.

We use the feature-learning technique of [11] to learn filters, which is effectively whitening and $k$-means (see the supplementary material for a thorough explanation). This procedure gives us a set of filters $\{f\}$ and a set of biases $\{b\}$, with which we can compute our feature channels $\{F\}$ as follows:

$$F^{(j)} = \max(0, (V \otimes f_j) + b_j) \tag{4}$$

Where $\otimes$ is pyramid filtering, as described earlier. We learn 26 channels, the same number as our "fixed" feature set, so that we can compare the effectiveness of both feature sets. We take a semi-supervised approach when learning features: for each tissue, we learn a different set of filters using only locations within 10 voxels of the tissue of interest. Examples of the channels we learn can be seen in Figure 2.

Note that our "adaptive" channels describe fundamentally different properties than our "fixed" channels. Our fixed channels describe the local distribution of a volume at a given location, orientation, and scale, while our adaptive channels describe the local distribution *of pyramid context features* at a given position, and as such they can describe non-local phenomena. An adaptive channel may learn to activate at voxels which are slightly to the left of some mass at a fine scale and distantly to the right of a much larger mass at a coarse scale, for example.

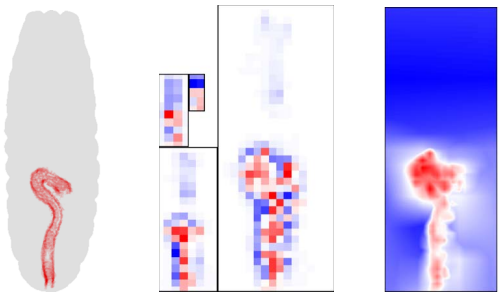
With our one "raw" channel, our 26 "fixed" channels, and our 26 "adaptive" channels, we can construct a feature vector for a voxel by computing pyramid context features for each channel at that voxel's location and concatenating those pyramid context vectors together (See Figure 2). This feature can be augmented by incorporating position information, as we will now demonstrate.

### 3.2. Position Features

Our imagery has been rotated to the "canonical" orientation used by the *Drosophila* community (see Figure 1(b)), and all volumes have been roughly registered to each other, which means that the absolute position of a voxel is informative. Our feature vector for a voxel's position is an embedding of the voxel's $(x, y, z)$ position into a multiscale trilinear spline basis. That is, we use trilinear interpolation to embed each voxel's position into a 3D lattice of control points, and we do this at multiple scales. Our resulting feature vector is mostly sparse, with values from 0 to 1, where the closer a position is to a control point determines how close that control point's bin is to 1 in the vector. We use a multiscale basis (different grids at different resolutions) to improve generalization: 4 lattices at different scales, with the coarsest having $(5 \times 2 \times 2)$ bins, and the finest having $(40 \times 16 \times 16)$ bins.

When extracting features for training, we construct these sparse position feature vectors using trilinear interpolation. Once we have trained a linear classifier (on a concatenation of our feature vector from Section 3.1 with these position features) we can evaluate the position part of the classifier by reshaping the weights into our multiscale lattice,

[2] in a slight abuse of our formalism in Section 2, instead of producing a feature channel and constructing a pyramid from that channel, we instead produce a pyramid from the volume, and then filter and rectify each scale of that pyramid independently. This works significantly better due to half-wave rectification being applied to the pyramid rather than the volume.



(a) A segmentation (b) Weights (pyr) (c) Weights (flat)

Figure 4. Because our volumes are in a canonical frame of reference, the absolute position of a voxel is informative. In 4(a) we have an embryo and a ground-truth annotation of a tissue, shown for reference. We then have the weights that our model learns for position for that tissue shown as a multiscale lattice (4(b)) and flattened to a single-scale volume (4(c)). Our multiscale representation allows our model to learn broad trends about position in coarse scales (such that the tissue is unlikely to occur at the top of the volume) while still learning fine-scale trends (like the shape of the tissue at the bottom of the volume). Weights are shown as max-projections, where red is positive, white is neutral, and blue is negative.

and collapsing that pyramid to be the same size as the input volume. This can be pre-computed, making evaluating this part of classification extremely fast: the collapsed pyramid of weights is just a per-voxel "bias". See Figure 4 for a visualization of a pyramid of learned weights for position, and of that pyramid collapsed to a volume.

### 3.3. Context

Given the features in Sections 3.1 and 3.2 we train a linear classifier (we use logistic regression) to produce a per-voxel prediction. This prediction is noisy, as we classify each voxel in isolation. We therefore construct a "two-layer" model which uses the prediction of the "single-layer" model to reason about the relative arrangement of the tissue, thereby adding information about context and self-consistency. We do this by making new "raw", "fixed" and "adaptive" features (Section 3.1) from the output of the single-layer model. We then learn a two-layer model which uses as its feature channels both the channels used in the first layer, and these new features built on the output of the first layer. See Figure 2 for examples of second-layer features and for a visualization of this two-layer architecture.

Of course, the output of our single-layer model is significantly more accurate on training-set volumes than on test-sets. This means that naively training a two-layer model on the output of the single-layer model can overfit drastically. To prevent this, when training single-layer models, we use leave-one-out cross validation on the training set to produce predictions for each training-set volume. This cross-validated output looks similar to the output of the model on the test-set. We train our two-layer model using these cross-validated predictions as input, which improves generalization on the test-set.

### 3.4. Post-Processing

Though our classification model can reason about context and self-consistency, its per-voxel predictions are still often noisy and incomplete at a fine scale. We therefore use a CRF-like technique to smooth and "inpaint" our predictions. We would like to smooth our predictions while still respecting intensity discontinuities in the raw input volume — that is, we want to smooth within tissue boundaries, but not across tissue boundaries. For this, we will use a joint-bilateral filter, where the predictions are smoothed in accordance with the intensity of the input volume.

We can efficiently apply a joint-bilateral filter using the bilateral grid [10]. We expand the output probabilities from our two-layer model to a 4-dimensional "grid", where each probability is embedded (or "splatted") with linear interpolation into one of three bins: low-intensity, medium-intensity, and high-intensity (bin centers are [8, 24, 48] ). The intensity bins are determined by the intensity of the raw volume while the quantity being filtered is the probability — hence the "joint" aspect of the bilateral filter. We then blur the 4D grid by convolving it with a 5-tap binomial filter in the three "position" dimensions and a 3-tap binomial filter in the "intensity" dimension. We then resample (or "slice") the smoothed 4D grid according to the linearly-interpolated volume intensity to produce a smoothed 3D volume. This procedure takes only a few seconds per volume. See Figure 5 for a visualization.

This joint-bilateral smoothing can be viewed as a single step of mean-field belief-propagation in a CRF, as in [17]. We experimented with complete belief-propagation, but found that only the first iteration contributed significantly to the output. This is probably because most tissues are usually so distant from the other tissues that the pairwise potentials have little effect.

(a) Input (b) Ground-truth (c) Raw Prediction (d) Smoothed Prediction

Figure 5. In 5(a) we have a cropped slice of an input volume, for which we have a ground-truth annotation of a tissue in 5(b). Our model produces the prediction in 5(c), which is often noisy and incomplete, so we use joint-bilateral smoothing to produce the smoothed prediction in 5(d), which propagates label information across the volume while respecting cell-boundaries.

### 3.5. Training

For each tissue we train a binary classifier using logistic regression, which we found to work as well as a linear SVM while having the benefits of being interpretable as probabilities and of introducing a non-linearity, which is important for our "two-layer" models. To train, we featurize each volume into a set of channels, and from those channels we extract many pyramid context features and corresponding position features, train a classifier, evaluate the classifier densely on each volume, and then mine for negatives (where a "negative" is a voxel labeled true with a probability less than $0.9$, or a voxel labeled false with a probability greater than $0.1$). We do 8 such bootstrapping iterations, after which most tissues converge. For our two-layer architecture, we do cross-validation on the training set, produce cross-validated predictions, produce features from those, concatenate those second-layer channels with our first-layer channels (and position), and then train on both with bootstrapping. We then apply our post-processing to the predicted output. We evaluate our results using per-voxel precision and recall, and report the average precision for each tissue. For our visualizations which require binary output such as Figure 1(c), we use the threshold which maximizes the F-measure of precision and recall on the training set.

## 4. Experiments

We demonstrate our semantic segmentation algorithm on fluorescence volumes of late-state *Drosophila* embryogenesis. We have a dataset of 28 volumes, each with a size of $454 \times 177 \times 185$, or nearly 15 million voxels. A *Drosophila* biologist annotated 8 biologically meaningful tissues, such as "left salivary gland" or "hindgut wall", and we split our annotated data into 14 training and 14 test volumes. The staining, imaging, and preprocessing of this imagery will be described in a later paper.

As mentioned previously, the large size and dimensionality of our data makes most pre-existing techniques difficult to use. Therefore, constructing good baseline techniques for comparison is very challenging. As one baseline we present an "oracle" segmentation technique: we use standard watershed segmentation techniques (threshold the volume, compute the distance transform, then compute the watershed transform) on the input volume to produce an oversegmentation of 10-25 thousand "supervoxels". At test-time we assign each supervoxel a prediction proportional to the fraction of the supervoxel that has been labeled in the ground-truth. This oracle technique gives us an upper-bound on the performance we should expect from super-voxel based semantic-segmentation techniques. This oracle performs poorly because so much detail is lost during the segmentation, demonstrating the value of our per-voxel classification technique. We attempted more sophisticated segmentation techniques such as those based on normalized-cuts, but these are intractable in our domain.

As a second baseline we present an "oracle" exemplar registration technique: for each test-set annotation we use iterative closest point to find an affine transformation from each training-set annotation to that test-set annotation, and then use the best-fitting training-set annotation as a per-voxel prediction by linearly interpolating the annotation into the test volume and blurring it by a $(1, 2, 1)$ binomial kernel. Because this prediction is produced by registering tissue annotations instead of actual tissues, this oracle technique serves as an upper bound on the performance we should expect from (affine) registration-based or correspondence-based techniques such as [15]. This oracle performs poorly, due to the heavy variation in each tissue and the fine-grained detail of cellular boundaries.

As a third baseline comparison, we use the well-known Histogram of Gradients [12] feature, generalized to volumetric data (gradients in 3D instead of 2D, 3D bins of size $4 \times 4 \times 4$, block-normalization, and $2 \times 2 \times 2$ cell arrangements), which we optionally augment with our position features from Section 3.2. Standard sliding-window detection with this 3D HOG feature is only tractable because of the severe pooling used in constructing the features — instead of 15 million voxels, we need only classify a quarter-million HOG features. But this comes with a cost, as these coarse features prevent us from producing per-voxel predictions. HOG is also limited in that it cannot incorporate contextual information without the feature vectors becoming intractably large. Of course, these limitations are exactly the motivation for our work.

Our other baselines are ablations of our technique, many of which are actually extremely similar to preexisting techniques. Pyramid context features on top of the raw input volume resemble the original use of Shape Context features [3], except that we use a soft rectangular Haar-like pooling instead of an expensive log-polar binning, and we use pyramid filtering to densely evaluate our classifier at every voxel instead of using correspondence for a sparse set of points. Our pyramid context features on top of our "fixed" feature channels also resemble Geometric Blur features [4, 5], except that instead of sampling a blurred signal is a log-polar arrangement, we sample a blurred signal is a rectangular Haar-like arrangement, and again use pyramid filtering instead of correspondence. That same model is also similar to Daisy features [23], but again made tractable using pyramid filtering. See Figure 3 for a comparison of these feature types. This comparison of our ablations to past techniques is generous, as pyramid context features and pyramid filtering are required to make all of these models tractable in our domain. Actually using standard sliding-window classification would take many hours per volume, making bootstrap-

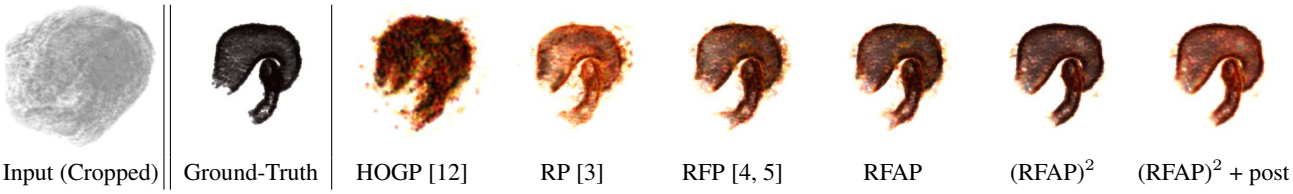



Figure 6. Some visualizations of the output of our model, and other models, on a test-set volume. In the first column we have the portion of the input volume containing the tissue, and in the second we have the ground-truth annotation of that tissue. The other columns are the output of various models, 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). Note that we process the entire volume, though we show a cropped view here. Many more similar figures and animations can be found in the supplementary material.

ping, evaluation, and experimentation nearly impossible.

In Table 1 we present the test set average precision for each model and each tissue. Model names are as follows: (1) is our "oracle" segmentation technique, (2) is our "oracle" exemplar warping technique, (3) is our HOG baseline, and (4) is (3) where features have been augmented with our position features. (20) is our complete model, and (5)-(19) are ablations of (20). (5)-(11) are single-layer models, where the model name indicates what features have been included: 'R' is the "raw" feature channel, 'F' is our "fixed" feature channels, 'A' is our "adaptive" feature channels, and 'P' is our position features. In models (12)-(14) we set $K$ (the number of levels in our Gaussian pyramids) to small values, to show the value of the coarse scales of our pyramid context features (in all other experiments, $K = 6$). (15)-(19) are our two-layer models, where we use the previously-described naming convention to indicate which features have been used for the second layer — so (RFAP)$^2$ uses all four feature types at both layers of the architecture. (20) is (19) with the post-processing filtering of Section 3.4 applied after classification. We also present Figure 1, which shows ground-truth and predicted labels for an entire test-set volume, Figure 6, which shows visualizations of the output of several models for a tissue, along with the ground-truth annotation and the (cropped) input volume, and Figure 7, which shows precision/recall curves for a subset of the models on two tissues. See the supplementary material for many more such visualizations.

Analyzing our results, several trends become clear. The oracle techniques, despite "cheating" by using the test-set labels, performs poorly. The HOG baseline does a very poor job because it cannot produce per-voxel predictions, and because it cannot reason well about context. Our position-only baseline shows that, even though our volumes are registered to each other, position information is not sufficient to solve this problem. Our ablations which resemble shape context and geometric blur features underperform our complete model, presumably because their input feature channels are impoverished. Both our "fixed" and "adaptive" feature channels improve performance, and so seem to contribute useful and complementary information. Our ablations in which our pyramid depths are limited perform poorly, as they are deprived of contextual information. Our two-layer model improves markedly over our single-layer model, and our post-processing helps greatly.

| Model | Tissue 1 | Tissue 2 | Tissue 3 | Tissue 4 | Tissue 5 | Tissue 6 | Tissue 7 | Tissue 8 | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| (1) Oracle Seg. | 0.745 | 0.791 | 0.781 | 0.677 | 0.762 | 0.818 | 0.783 | 0.787 | 0.768 |
| (2) Oracle Warp | 0.485 | 0.597 | 0.592 | 0.468 | 0.464 | 0.746 | 0.443 | 0.476 | 0.534 |
| (3) HOG [12] | 0.249 | 0.252 | 0.256 | 0.101 | 0.173 | 0.470 | 0.157 | 0.250 | 0.239 |
| (4) HOGP [12] | 0.417 | 0.339 | 0.345 | 0.204 | 0.431 | 0.545 | 0.337 | 0.371 | 0.374 |
| (5) P | 0.227 | 0.200 | 0.247 | 0.127 | 0.237 | 0.261 | 0.212 | 0.249 | 0.220 |
| (6) R [3] | 0.427 | 0.361 | 0.371 | 0.280 | 0.349 | 0.717 | 0.709 | 0.759 | 0.496 |
| (7) RP [3] | 0.705 | 0.688 | 0.691 | 0.425 | 0.679 | 0.848 | 0.818 | 0.846 | 0.712 |
| (8) RFP [5] | 0.843 | 0.878 | 0.867 | 0.720 | 0.851 | 0.939 | 0.918 | 0.925 | 0.868 |
| (9) RAP | 0.857 | 0.863 | 0.859 | 0.736 | 0.887 | 0.943 | 0.927 | 0.933 | 0.876 |
| (10) RFA | 0.860 | 0.890 | 0.894 | 0.783 | 0.889 | 0.953 | 0.935 | 0.939 | 0.893 |
| (11) RFAP | 0.869 | 0.893 | 0.890 | 0.775 | 0.898 | 0.952 | 0.937 | 0.941 | 0.894 |
| (12) RFAP, K=1 | 0.781 | 0.768 | 0.769 | 0.590 | 0.814 | 0.904 | 0.895 | 0.901 | 0.803 |
| (13) RFAP, K=2 | 0.828 | 0.848 | 0.845 | 0.694 | 0.864 | 0.930 | 0.919 | 0.918 | 0.856 |
| (14) RFAP, K=3 | 0.848 | 0.890 | 0.887 | 0.778 | 0.885 | 0.948 | 0.928 | 0.933 | 0.887 |
| (15) RFAP × RP | 0.880 | 0.903 | 0.897 | 0.793 | 0.913 | 0.958 | 0.938 | 0.946 | 0.904 |
| (16) RFAP × RFP | 0.887 | 0.909 | 0.905 | 0.810 | 0.925 | 0.965 | 0.939 | 0.948 | 0.911 |
| (17) RFAP × RAP | 0.894 | 0.917 | 0.912 | 0.818 | 0.932 | 0.964 | 0.941 | 0.950 | 0.916 |
| (18) RFAP × RFA | 0.891 | 0.916 | 0.910 | 0.837 | 0.925 | 0.967 | 0.947 | 0.952 | 0.918 |
| (19) (RFAP)$^2$ | 0.894 | 0.914 | 0.912 | 0.825 | 0.934 | 0.967 | 0.942 | 0.953 | 0.918 |
| (20) (RFAP)$^2$ + post | 0.914 | 0.934 | 0.933 | 0.865 | 0.945 | 0.975 | 0.947 | 0.958 | 0.934 |

Table 1. Test-set average precisions for our model (20), several ablations of our model (5-19, some of which resemble past techniques), a baseline technique adapted to volumetric data (3-4), one "oracle" technique based on oversegmentation (1), and another "oracle" based on exemplar-based registration. We report APs for the 8 different tissues in our dataset, and the mean AP across all tissues. See the text for a description of each technique.

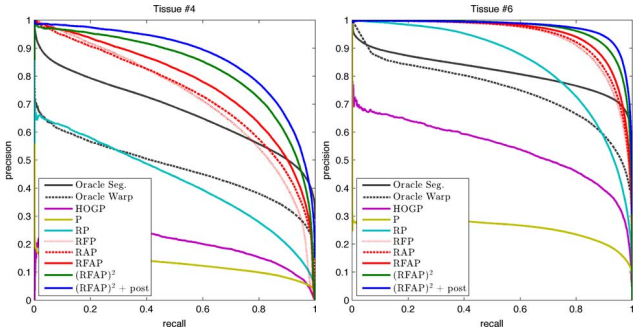



Figure 7. Precision/recall curves for different models on our entire test set, for one specific tissue. On the left we have the hardest tissue in our dataset (the one for which our model and the baselines performs worst) and on the right we have the easiest. See the supplementary material for the AP curves for all tissues.

## 5. Conclusion

We have presented an algorithm for per-voxel semantic segmentation, demonstrated on 3D fluorescence microscopy data of *Drosophila* embryos. The size and high-dimensionality of our data renders most existing techniques intractable or inaccurate, while our technique produces very accurate per-voxel segmentations extremely efficiently — hundreds of times faster than existing techniques. At the core of our algorithm is our novel pyramid context feature, which is not only a powerful descriptive representation, but is designed such that exact per-voxel linear classification can be made extremely efficient. We have demonstrated our model's efficiency both empirically, through experimentation, and analytically, through complexity analysis. For our semantic segmentation algorithm, we have introduced three feature types — a standard feature set that uses oriented edge information, a novel feature set produced by applying feature-learning to pyramid context features, and a feature which encodes absolute position information. By learning classifiers on top of pyramid context features based on these channels we can produce per-voxel segmentations, which can be improved with contextual information by "stacking" our models and using the output of one layer as input into the next. We have also presented a CRF-like post-processing technique for improving our output using joint-bilateral filtering.

Besides advancing computer vision research, our work has the added benefit of tackling a crucial and unsolved problem in *Drosophila* research — that of automatically constructing an atlas of embryo morphology. By efficiently and accurately producing semantic segmentations of tissues from volumetric data, we enable real, breakthrough biological research at a large scale.

## References


[1] B. Andres, T. Kroeger, K. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. Hamprecht. Globally optimal closed-surface segmentation for connectomics. *ECCV*, 2012.

[2] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. *CVPR*, 2012.

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

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

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

[6] X. Boix, J. Gonfaus, J. van de Weijer, A. Bagdanov, J. Serrat, and J. Gonzalez. Harmony potentials: Fusing global and local scale for semantic image segmentation. *IJCV*, 2012.

[7] J. Campos-Ortega and V. Hartenstein. The embryonic development of drosophila melanogaster. *Springer*, 1997.

[8] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. *ECCV*, 2012.

[9] J. Carreira, F. Li, and C. Sminchisescu. Object recognition by sequential figure-ground ranking. *IJCV*, 2012.

[10] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *SIGGRAPH*, 2007.

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

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

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

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge. http://pascallin.ecs.soton.ac.uk/challenges/VOC/.

[15] C. C. Fowlkes, C. L. Luengo Hendriks, S. V. E. Keränen, G. H. Weber, O. Rübel, M.-Y. Huang, S. Chatoor, A. H. DePace, L. Simirenko, C. Henriquez, A. Beaton, R. Weiszmann, S. Celniker, B. Hamann, D. W. Knowles, M. D. Biggin, M. Eisen, and J. Malik. A quantitative spatiotemporal atlas of gene expression in the drosophila blastoderm. *Cell*, 133(2), 2008.

[16] A. Hiang, C. Lin, C. Chuang, C. H., C. Hsieh, Y. C., S. C., J. Wu, W. G., and Y. Chen. Three-dimensional reconstruction of brain-wide wiring networks in drosophila at single-cell resolution. *Curr. Biol.*, 21, 2011.

[17] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*, 2011.

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2006.

[19] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. *ICCV*, 1999.

[20] F. Long, H. Peng, X. Liu, S. Kim, and E. Myers. A 3D digital atlas of C. elegans and its application to single-cell analyses. *Nature Methods*, 6(9), 2009.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[22] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *JOSA A*, 1990.

[23] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *TPAMI*, 2010.

[24] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *TPAMI*, 2010.

[25] A. Vazquez-Reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfiste. Segmentation fusion for connectomics. *ICCV*, 2011.

[26] S. Vitaladevuni and R. Basri. Co-clustering of image segments using convex optimization applied to EM neuronal reconstruction. *CVPR*, 2010.