SCALPEL: Segmentation CAscades with Localized Priors and Efficient Learning

David Weiss
University of Pennsylvania
Philadelphia, PA
djweiss@cis.upenn.edu

Ben Taskar
University of Washington
Seattle, WA
taskar@cs.washington.edu

Abstract

We propose SCALPEL, a flexible method for object segmentation that integrates rich region-merging cues with mid- and high-level information about object layout, class, and scale into the segmentation process. Unlike competing approaches, SCALPEL uses a cascade of bottom-up segmentation models that is capable of learning to ignore boundaries early on, yet use them as a stopping criterion once the object has been mostly segmented. Furthermore, we show how such cascades can be learned efficiently. When paired with a novel method that generates better localized shape priors than our competitors, our method leads to a concise, accurate set of segmentation proposals; these proposals are more accurate on the PASCAL VOC2010 dataset than state-of-the-art methods that use re-ranking to filter much larger bags of proposals. The code for our algorithm is available online.

1. Introduction

In this paper, we focus on object localization and segmentation. A common and highly successful approach is to generate a large set of (possibly overlapping) proposed segmentations, sometimes called a "bag of segments" or "soup of segments." These proposals can then be evaluated by a more complex model to determine a final set of localized and segmented objects in the image. These proposal-based methods have proven very useful for both object detection/recognition tasks (e.g. [20]) as well as image/object segmentation (e.g. [1, 11, 22]).

A typical processing pipeline used by several state-of-the-art bag of proposal methods [7, 5, 12] is diagrammed in Figure 1 (Bottom half). Given an input image, many seed regions of interest are generated. Next, a pixel-wise segmentation model is solved using graph-cut to find many solutions using several different parameter settings. Because graph-cut is used as inference, the features used in the segmentation model must be pairwise and sub-modular; pairwise features primarily use bottom-up cues such as boundaries, texture, etc. These pairwise features are not sufficient to discriminate between full objects and partial segmentations, so many proposals must necessarily be generated per seed, and many seeds must be sampled. Therefore, the final step of the process involves learning a re-ranking classifier to filter the fixed set of segment proposals using features computed over the entire region, i.e. normalized cut energy or Gestalt features such as shape moments.

In this work, we propose incorporating region features normally reserved for a re-ranker directly into the segmentation process. This allows us to be far more efficient in terms of the number of proposals generated, as our method can provide similar or better accuracy as state-of-the-art re-ranking based systems with only a single proposal per region of interest. Because we can evaluate features such as normalized cut energy during the segmentation process, our procedure can find the right balance between filling out a given region and finding a segmentation that has object-like properties as a whole. Given a prior belief corresponding to an object in the image, our approach is more likely to get the segmentation right the *first time*, without needing to generate multiple guesses.

However, incorporating region features comes at a price: we must forego using efficient graph-cut algorithms to produce our segmentation proposals. To retain efficiency, we adopt instead a greedy superpixel selection algorithm. While the concept of segmentation through greedy superpixel selection is at least a decade old (e.g. [17]), we provide two main innovations that are necessary in order for such a method to outperform state-of-the-art Conditional Random Field (CRF) approaches (Figure 1). Specifically, we incorporate high-level information about the scale, probable layout, class of the object, and the current stage of segmentation into the procedure. This is because there are intrinsic variations across and within objects that will affect whether or not a given feature is useful during the greedy segmentation process. For example, the usefulness of color and

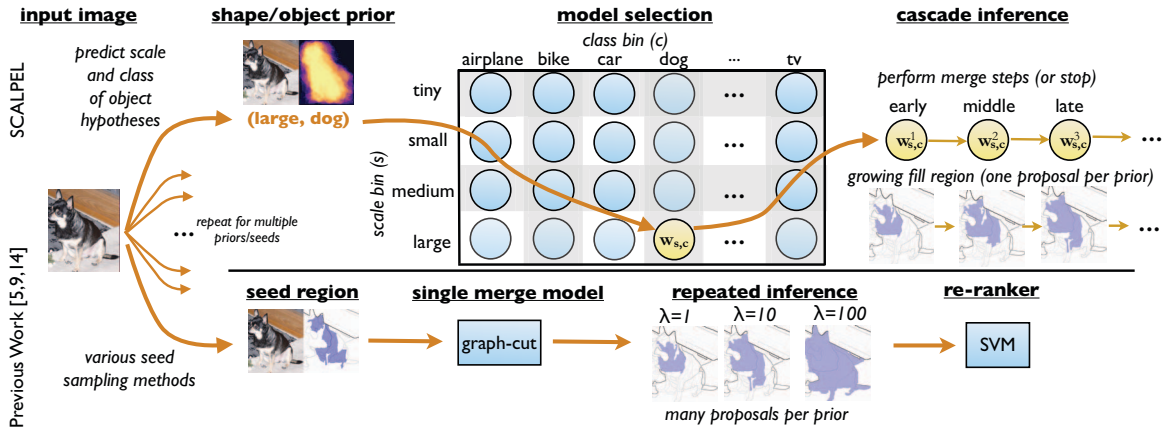

Figure 1. **Top:** Overview of SCALPEL segmentation. The input is a shape prior annotated with class and size information (automatically generated from the prior generator.) The class and size are used to select a scale and class specific cascade model $\mathbf{w}_{s,c}$ from a lookup table. The cascade greedily grows a fill region (initialized from the shape prior), with each sub-model adding up to a fixed number of segments before passing the region to the next level of the cascade, stopping whenever any level finds no superpixel candidate scoring above zero. **Bottom:** Overview of previous graph-cut and re-ranking based pipelines.

texture information from the prior depends on whether or not the object class has consistent color and texture, as well as if the object is too large or small to get good estimates of color and texture. Similarly, segmenting a large object requires *ignoring* interior boundaries early on in the greedy selection process, yet *respecting* exterior boundaries once the object has been fully segmented in the later stage of the process.

We model these variations by (i) learning scale and class specific models where different weight vectors are used for different *(scale, class)* combinations, and (ii) learning a *cascade* of selection models in which different stages of the segmentation process have different parameters. Unlike a fixed model, the cascade is capable of learning to ignore boundaries early on in the process yet use them as a stopping criterion once the object has reached a certain size. When paired with a novel system that generates better localized shape priors than our competitors, our method leads to a concise, accurate set of roughly 650 proposals per image *without* any re-ranking; moreover, these proposals are more accurate on the PASCAL VOC2010 dataset than other state-of-the-art methods [5, 7]. We call our approach **S**egmentation **Ca**scades with **L**ocalized **P**riors and **E**fficient **L**earning, or SCALPEL. To summarize, the contributions of this work are as follows.

1. **Region features during segmentation.** We demonstrate that it is feasible to incorporate features normally reserved for re-ranking directly into the segmentation process, using a simple greedy method for superpixel selection.
2. **Efficiently trained segmentation cascades.** We show how to efficiently train a set of models for greedy superpixel selection using standard SVM solvers.
3. **Accurate greedy segmentation.** We show how to significantly increase the performance of the model by learning separate models for objects of different scale and class. We learn to infer these properties using a simple localized shape prior generation scheme that localizes objects with higher recall than either purely bottom-up or top-down methods. Notably, we outperform graph-cut based approaches on the challenging PASCAL VOC2010 Segmentation dataset.

2. Related Work

Several previous works have attempted to form segmentations of objects in the image given a detection bounding box [6, 22, 10, 14, 15]. [6] and [22] both learn several shape priors using the root or parts of the DPM [9], whereas we learn hundreds of holistic shapes from a fine-grained clustering in mask pixel space. Furthermore, [6, 22] trust the class assignments of the detections; we use bottom-up bounding boxes with no class information to increase recall, and instead generate shape predictions based on the content of the bounding boxes. Most importantly, where [14, 10] simply transfer a matched object mask and [6, 15] learn to segment on pixels using graph-cut and sub-modular edge features, we learn to greedily segment on superpixels with *arbitrary* features, a *cascaded* weight vector, and with a set of class- and scale-specific models.

Both [7, 6] incorporate learning into their segmentation method by learning either a set of unary scores for graph-cut [7] based on harvested region pairs or by directly using max-margin structured learning with graph-cut as inference [6]. While [6] combines bottom-up and top-down image

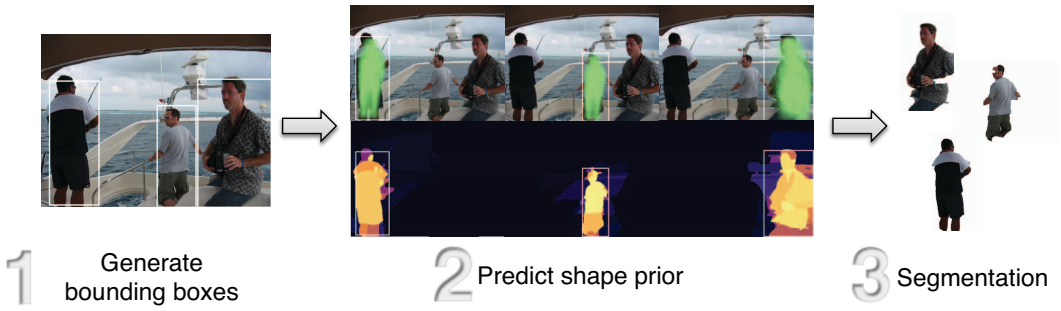

Figure 2. Overview of entire SCALPEL processing pipeline. (1) Roughly 900 Bounding boxes per image are generated from several different sources combining bottom-up and top-down information. (2) Top: Each box is evaluated by a shape classifier (Section 4) that provides a rough estimate of the shape of an object inside the box. Bottom: These estimates are integrated over superpixels to form a localized shape prior. (3) Superpixels are greedily selected using a cascaded segmentation model to form the final output (Section 3).

cues as we do, these cues are transferred from a nearby training example, and not via a learned shape prior as in our approach. Furthermore, these approaches are again limited to sub-modular edge weights and graph-cut as inference, while we in contrast learn cascaded weights on features computed over arbitrary groups of superpixels that are not required to be sub-modular.

Our greedy inference procedure is superficially similar to the Maximal Weighted Clique (MWC) scheme used by [11]. However, our goal is to group super-pixels into a single coherent region, not to produce a tiling of segments that cover coherent regions over the entire image; we stop agglomerating superpixels when the score no longer exceeds a desired threshold. More importantly, our cascaded weight vector allows for the scoring function to change as inference proceeds, features can be computed over arbitrary groups of superpixels (not only pairwise), and we use the localized shape priors to seed our method and guide inference. Rather, our segmentation approach is more related to the greedy MCMC inference used by [17], again with the addition of shape priors and our modeling innovations. Our cascade approach is most similar to that developed independently by [18], but we incorporate high-level information rather than taking a purely bottom-up approach.

Finally, we note that our shape priors are inspired by mask transfer approaches to segmentation, i.e. extracting descriptions of object shape from training examples and matching these to regions in a test image. Class-specific matching typically requires detecting objects and then adapting a segmentation mask to the detected box from either a single training exemplar [14] or cluster of exemplars [10]; alternatively, fragments of the test image can be matched with fragments of objects of known shape and class [4, 3]. Category-independent matching ignores the class label of training exemplars and attempts to match either regions [12], windows [13], or entire images [19], without regard to detected objects. We take a middle-ground approach; we combine predictions from both category-independent and class-specific methods to form the input to our segmentation model.

3. Learning to segment with SCALPEL

Given a prior belief about an object in an image, our goal is to find a selection of superpixels that both match the prior and have excellent support from image cues. Here, we focus on the segmentation; we discuss the prior generation in Section 4. One price we pay by incorporating region-based features into the segmentation process is that pixel-wise segmentation becomes prohibitively expensive. Therefore, we opt to perform segmentation at the superpixel level, using the output of gPb-owt-ucm [2] with 200 superpixels.

3.1. Segment selection algorithm

We first describe a greedy segment selection algorithm without a cascade; we will then extend the algorithm to the cascaded setting. Intuitively, our algorithm begins with a single superpixel and then repeatedly adds neighboring superpixels to the set until a stopping criterion is reached. Let $S(x) = \{1, \ldots, 200\}$ be the set of superpixels for an input x. We represent a filled-in object mask as a subset of superpixels that we turn "on." We represent a labeling of x as a binary vector y, where $y_i = 1$ if superpixel i is included in the object and $y_i = -1$ otherwise. Because the greedy inference algorithm selects superpixels sequentially, we also define a *selection order* z to be an ordered subset of $S(x)$ indicating the order in which superpixels were selected by the greedy algorithm.

We next define our features in terms of the decisions made by the greedy inference scheme. Given a selection order z and a candidate superpixel s, the algorithm computes features $\Delta\mathbf{f}(x, z, s)$ that measure the change in region properties when s is selected as the next element. Note that these difference features are typically very computationally efficient, as only the changes in region properties by adding s to z need to be computed.

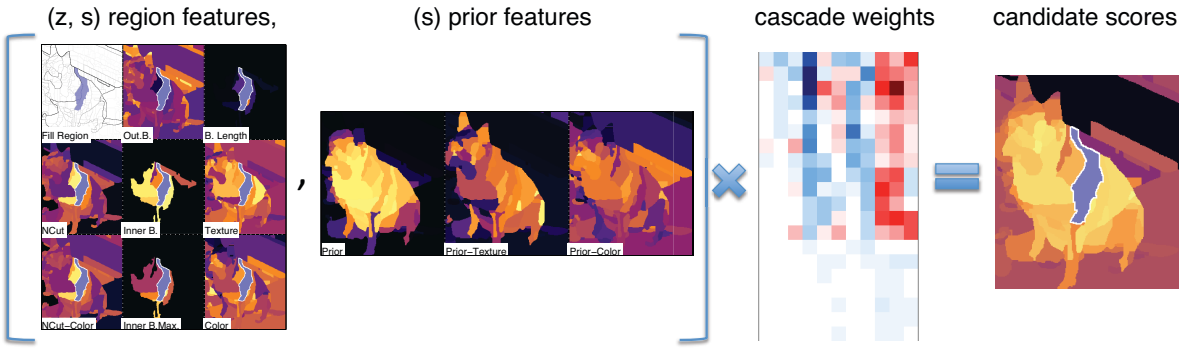

Figure 3. Feature overview and scoring. The feature vector $\Delta\mathbf{f}(x, y, s)$ consists of 8 region features and 3 unary features. Above, we show each feature channel for the segmentation of a dog instance. y is visualized for reference as a fixed blue-colored region; the superpixels are then colored according to the value of each feature computed w.r.t this fill region. The features are scored according to a single selected row of the cascaded weight vector to produce the final scores over candidate superpixels on the right.

Given a weight vector $\mathbf{w}$, feature generating function $\Delta\mathbf{f}$, and initialization s_1, the inference procedure greedily optimizes the following linear scoring function:

$$z^\star(\mathbf{w}, x) = \underset{z:z_1=s_1}{\operatorname{argmax}} \sum_i \mathbf{w}^\top \Delta\mathbf{f}(x, z_{1:i}, z_{i+1}), \quad (1)$$

where $z_{1:i}$ is the first i elements of z. Inference proceeds as follows. Let $z^{(t)}$ be the selection order so far at iteration t, where $z^{(1)} = s_1$. We define the best next candidate $s^{(t)}$,

$$s^{(t)} = \underset{s\in N(z^{(t)})}{\operatorname{argmax}} \mathbf{w}^\top \Delta\mathbf{f}(x, z^{(t)}, s), \quad (2)$$

where $N(z)$ are the neighboring superpixels to those already in the selection order z. In other words, $s^{(t)}$ is the neighboring superpixel with largest score according to the current selection order $z^{(t)}$. We then define the greedy update to the selection order $z^{(t)}$ at step t to update only if the estimated change in score is positive:

$$z^{(t+1)} \leftarrow \begin{cases} z^{(t)} & \text{if } \mathbf{w}^\top \Delta\mathbf{f}(x, z^{(t)}, s^{(t)}) < 0 \\ z^{(t)} \cup s^{(t)} & \text{otherwise.} \end{cases} \quad (3)$$

We now extend the greedy algorithm to a *cascaded* setting in a straightforward fashion. Specifically, we define a series of K weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_K$. Inference then proceeds in a stage-wise fashion, where stage k uses $\mathbf{w}_k$ to either stop inference or select one or more additional superpixels before passing to the next stage (Figure 1). Specifically, we substitute $\mathbf{w}$ in (3) with $\mathbf{w}_{\ell(x,z^{(t)})}$, where $l(x, z^{(t)}) \mapsto \{1, \ldots, K\}$ is a *cascade lookup function* that defines the schedule of the cascaded inference. For example, one simple lookup schedule is to evenly divide the range of inference steps each stage of the cascade, using $\ell(x, z) = \lceil |z|K/|S(x)| \rceil$. In Section 5, we discuss a more useful lookup schedule based on our shape priors (Section 4).

3.2. Learning the Cascade

We now turn to learning the weights $\mathbf{w}_1, \ldots, \mathbf{w}_K$. We use the shorthand $\{\mathbf{w}_k\}$ to refer to the set of weights jointly, and we use $\psi_j(z, s) = y_s^j \mathbf{w}_\ell(x^j, z) \cdot \Delta\mathbf{f}(x^j, z, s)$ to be the SVM-style margin of selecting superpixel s given selection z on input x^j. Our training procedure is simple, and learns the cascade in a bottom-up fashion. We treat each selection step (3) as a binary classification problem as follows. Recall that y_s is the label of superpixel s; we then define the following standard max-margin hinge loss objective, summed over a training set $\{(x^j, y^j)\}_{j=1}^n$ of n examples:

$$L(\{\mathbf{w}_k\}) = \sum_{j=1}^{n} \sum_{z, s\in N(z)} \max\{0, 1 - \psi_j(z, s)\}, \quad (4)$$

where we sum over all selection orderings z and possible selections s. Intuitively, this objective states that all selections that correct select the next superpixel, regardless of context, should score positively, while all incorrect selections should be rejected.

Unfortunately, there are far too many selection orderings for optimization of (4) to be practical. Instead, we iteratively approximate the sum in (4) with samples from a single selection ordering z^j:

$$L'(\{\mathbf{w}_k\}) = \sum_{j=1}^{n} \sum_{i, s\in N(z^j_{1:i})} \max\{0, 1-\psi_j(z^j_{1:i}, s)\}. \quad (5)$$

Note that given z^j, we can optimize (5) using standard SVM solvers such as [8]. We found that an iterative procedure to choose z^j worked well in practice: we first set $z^j = \{1, \ldots, S(x^j)\}$ and solve for $\{\mathbf{w}\}_{1:K}$. We then fix $\mathbf{w}_1$, set $z^j = z^\star(\{\mathbf{w}\}_{1:K}, x^j)$ and repeat, fixing $\mathbf{w}_2$ on the next iteration, and so forth until all K models have been learned.

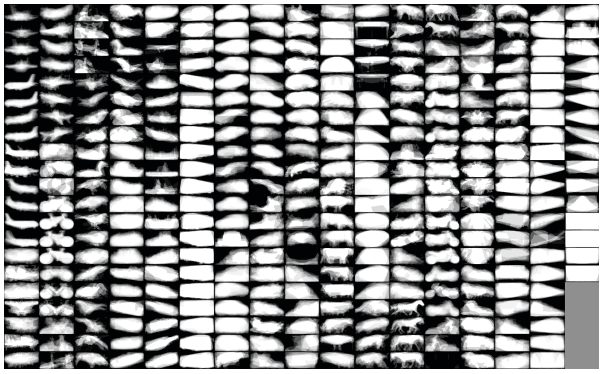

Figure 4. Example of the soft object mask dictionary learned for a single aspect ratio, sorted by class. Each sub-image is the average resized mask of all objects in that cluster.

3.3. Class- and Scale- Specific Models

Class- and scale-specificity. While many of the features we use are generic, there are intrinsic variations across objects that will affect whether or not a given feature is useful to greedily segment that object. Although the cascaded model introduced in the previous section can learn to discount features early on in the segmentation process and change weights as inference proceeds, it cannot handle *a priori* variations between objects due to either object size or object category. We propose a simple yet effective scheme to model these variations: we divide objects in the training set into 5 different scales and by each of the 20 classes in the PASCAL dataset, and for each unique *(scale, class)* pair, we learn an entirely separate segmentation model. At test time, we use the area of the target bounding box to determine the scale bin and the output of the shape classifier to determine the appropriate class bin, and run the selected model accordingly (Figure 1). Note that even if the class and scale predictions are incorrect at test time, selecting a different model for each prior is a useful way to generate a diverse set of proposals from a pool of priors.

Features. The segmentation model uses 8 features computed on groups of segments and 6 unary features for a total of 14 features. These features are summarized and visualized for a particular example in Figure 3. The unary features are computed once for each superpixel s, while the region features are computed during inference, relative to a fixed already-filled region z and a candidate superpixel s. The first two region features are the difference in normalized cut energy if s is added to z using both boundary strength and color similarity as the cut edge weights. The next features are computed on the boundaries: the strength of the exterior boundary of s w.r.t. z, the strength and max strength of the interior boundary of s w.r.t. z, and the total boundary length. The final two region features are the average similarity between superpixels within z and the candidate s in terms of color and texture. The six unary terms are the localized shape prior and the output of two logistic regression models trained to differentiate the prior from the background using texture and color respectively, plus three bias terms: a generic bais, a term for the size of the superpixel, and a term for the ratio of the boundary to the area of the superpixel.

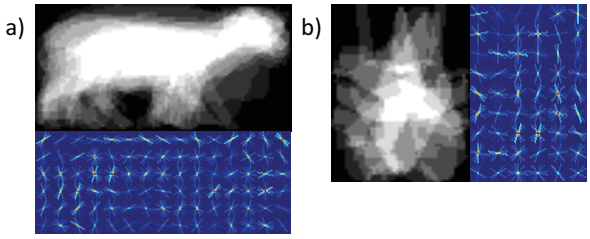

Figure 5. Examples of HoG based shape classifiers. While (a) is a typical informative shape prior from the *sheep* class, the cluster of *aeroplane* objects in (b) are mis-aligned, do not provide an informative prior, and are discarded at run-time.

4. Generating Localized Shape Priors

From object hypotheses to localized priors. In this section, we now explain how we generate the input to the segmentation pipeline (Figure 2). First, we sample bounding boxes from three different publicly available bounding box generation methods: purely bottom-up boxes from the segmentation hierarchy of gPb-owt-ucm [2], category-independent boxes from [16], and purely top-down class-specific boxes from [9]. (For more information on the bounding boxes, see the supplement.) We pool the boxes together and discard the class information provided by [9] to form our object hypotheses. Our next step is to predict a soft object mask (Figure 4) for each box, which we in turn use to compute a localized shape prior for the image. The final input to segmentation algorithm is therefore a pool of shape priors annotated by *(scale, class)* pairs, where the area of the bounding box is used for the scale of the object, and the class is taken from the predicted object mask.

Learning a soft mask dictionary. In order to predict soft masks for each bounding box, we first need a dictionary to define the space of possible soft masks (Figure 4). Using the annotations provided by [4], we first cluster all objects in the PASCAL VOC2011 training set into five aspect ratio clusters using K-means. We then extract the binary mask for each object and resize into a low-resolution thumbnail. The use of thumbnails ensures that minor variations in shape will not significantly change distance in mask pixel space. Using [21], we perform hierarchical K-means clustering in mask pixel space for every unique *(aspect ratio, class)* pair that exists in the dataset, producing 92 cluster trees. Finally, we pool all of the mask clusters for a given aspect ratio into a single set, yielding roughly 350 shapes per aspect ratio and a total of 1428 shapes.

Method	IoU	Recall	Covering
Priors Only	71.1	80.7	80.4
Object proposals [7]	71.2	82.5	79.5
CPMC [5]	71.4	79.9	82.8
SCALPEL	**73.1**	**82.9**	**82.9**

Table 1. Segmentation results on VOC2010 validation set. Note that the SCALPEL priors are already competitive with state-of-the-art; SCALPEL outperforms [7] and [5] in every metric.

Learning a soft mask classifier. Given a bounding box, we need to choose the soft mask that best matches the object inside the box. In the interests of generalization and efficiency, we opt not to use nearest-neighbor methods, and instead learn a linear SVM classifier using Histogram of Gradients (HoG) features to differentiate between exemplars in each soft mask cluster. We use the LIBLINEAR [8] software package to train a linear multi-class SVM classifier, and we choose the regularization parameter C of the SVM as well as the size of the HoG cells using cross-validation on a development set. After the first round of training, we harvest false-positives from the bounding box pools on the training set and introduce them as examples of an additional negative class for a second round of training (Figure 5).

Localization with superpixels. The predicted soft object mask is often only roughly aligned with the object (e.g. Figure 2), and leaks into the background. To fix this, we integrate the soft mask over underlying superpixels and normalize by the area of each superpixel. This largely eliminates bleeding into the background when the background consists of large superpixels and the mask at least partially covers the object. We also discard soft object masks that suffer from misalignment in the corresponding cluster by throwing out any predicted mask classifications where the average soft mask accounts for less than 40% of the hypothesized bounding box (Figure 5).

5. Segmentation Results

Experimental design. We train our method using the PASCAL VOC2012 trainseg set, we evaluate our approach on the VOC2010 valseg set. To develop our algorithm, we used the images in the set $\{\text{valseg2012} - \text{valseg2010}\}$.

Cascade schedule. The schedule should ideally be aware of not simply how many superpixels have been selected, but how much of the *object* has been segmented, so that different weights can be used for early vs. late decisions in the segmentation procedure. Although we do not know at test-time how much of the object is covered, the shape priors are a useful substitute; therefore, in practice, we set $\ell(x, z) = \lceil \sum_i p(x, z_i)K \rceil$, where $p(x, j)$ is the fraction of the shape prior covered by superpixel j for input x. Furthermore, for scale-specific cascades, we choose K based on the scale of the target object, as selecting even a single superpixel will use up significant percentages of the prior for smaller objects. We use $K = \{1, 2, 4, 8, 16\}$ for each scale bin respectively. For non-scale specific cascades, we use $K = 16$.

Specific?	K	IoU	Recall	Covering	Avg
—	1	72.1	81.0	82.0	78.4
—	16	72.5	81.4	82.5	78.8
Scale	1	72.4	81.4	82.1	78.6
Scale	16	72.7	82.0	82.3	78.9
Class	1	72.4	81.6	82.7	78.9
Class	16	**73.2**	82.4	**83.1**	79.5
Class&Scale	1	73.0	82.6	82.7	79.4
Class&Scale	16	73.1	**82.9**	82.9	**79.6**

Table 2. Effect of model complexity on performance for various SCALPEL variants. Adding specificity and cascaded weights always leads to an increase in performance. Overall, superior average performance across the three metrics is achieved with class- and scale-specific cascades.

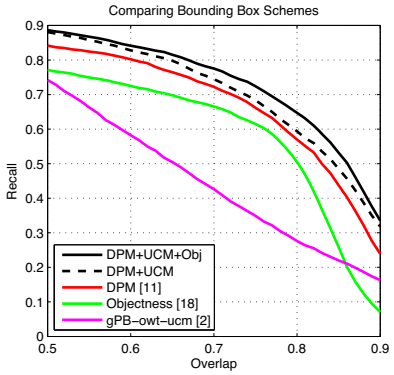

Figure 6. Comparison of bounding box strategies. Recall is computed for a fixed number of boxes.

Implementation/Run-time. To seed our segmentations, we use the single superpixel with highest localized prior score. For features, we compute dense PHoW using [21] for texture descriptors and the (L,a,b) colorspace as our color descriptors. The descriptors are discretized into a codebook on the training set and we use the inner product of histograms to compute similarities in texture and color between pairs of superpixels. In our region features, all similarities are weighted by the size of the originating superpixel to aggregate similarities. Our system runs at speeds comparable to other state-of-the-art systems; after roughly 4-5 min of preprocessing to compute gPb-owt-ucm and features per image, prior prediction takes roughly 30s and segmentation takes 2-4 min per image, running unoptimized MATLAB code on a 2.8 Ghz Opteron machine.

Evaluation. For each proposed segment and ground truth object in an image, we compute the *overlap* score, which is the sum of the intersection of the two masks divided by the union (abbreviated IoU). To evaluate a pool of segments

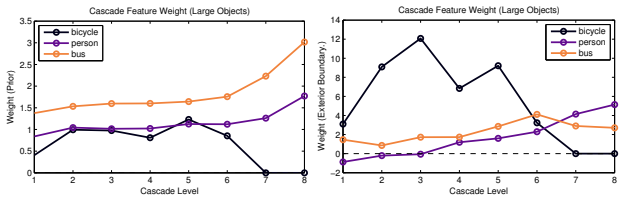

Figure 7. Diversity of weight vectors for large object cascade for different classes. Left: Weight of shape prior. The prior is more important for easily identifiable shapes (buses) than for more complex objects (people and bicycles). Note that bicycles do not learn any feature weights for the later stage of the cascade, instead always stopping early. Right: Weight of exterior edge feature. Exterior edges are far more important for bicycles than for buses and people; the people cascade learns to ignore exterior edges until the object is mostly segmented.

with respect to a given object, we report the best overlap score across all segments. As we are interested in the precision and recall of the segment pools, we compute average best overlap across all objects in the test set, as well as recall percentage at the standard 0.5 overlap threshold. We also follow [5, 2, 12] and report the average *covering* of each image in the test set. For a given pool of segments and objects, the covering metric is the average best overlapping score between ground-truth and proposed segments, weighted by the number of pixels in each object. Because the covering penalizes incorrect segmentation of large objects greater than small objects, we also investigate the average overlap as a function of object size.

Baselines. We compare several variations of our method to the publicly available implementation of CPMC, which we ran with default parameters. We also use precomputed bags of ranked proposals provided by [7]. We calibrated each method to output roughly 650 proposals per image, which is the number of proposals produced with the CPMC default parameters (we did not run the CPMC re-ranking step.) Finally, since we use the same error metric and the same evaluation data, we also compare to the published results of [12], as there was no publicly available implementation.

Localized Prior Quality. We first evaluated the quality of the bounding boxes and localized priors themselves. We first compared our mixed bounding box sampling approach against sampling 900 boxes of each method individually (Figure 6), and found that our method greatly increases recall compared to any individual method. Next, to generate a proposed segmentation for each prior, we greedily selected superpixels to obtain a segmentation with highest overlap with the soft mask. We compare to the reported numbers from Shape Sharing [12] for their neighbor-search based priors. We find that our priors, while similar in number (658 vs. 608), have significantly higher covering than the Shape Sharing priors (80.4% vs 77.0%). Note that we would expect our priors to be more informative, as our method leverages category-specific information, while Shape Sharing is entirely category-independent.

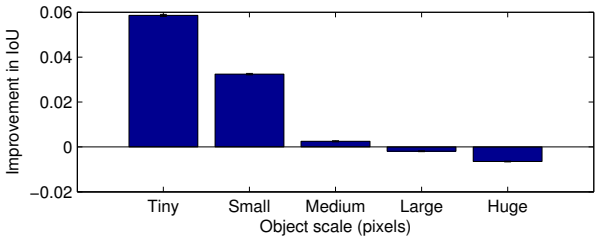

Figure 8. Improvement of SCALPEL over CPMC grouped by scale of the object. The gains of SCALPEL come in the difficult tiny and small objects.

Segmentation Proposal Quality. We evaluate several variants of our method against several state-of-the-art baselines. First, we investigated the contribution of the various techniques we applied to make the greedy inference procedure robust to variations within and across objects (Table 2). We find that both cascades and class- and scale-specific models are important, effective means of improving the performance of the greedy inference scheme. Based on our development set, we choose the final variant including cascades and both specificities to represent SCALPEL in a comparison against state-of-the-art baselines.

We compared SCALPEL to Object Proposals [7] and CPMC [5]. Note that Object Proposals initially generates many more proposals, but uses a separately trained re-ranker to reduce their number. While CPMC sacrifices recall for covering and Object Proposals sacrifices covering for recall, SCALPEL outperforms both on IoU, recall, and covering simultaneously. We also compared favorably to Shape Sharing [12], which uses 1448 proposals per image and achieves 84.3% covering; by proposing two shape priors per bounding box, we can increase our proposals to 1456 and achieve 84.4% covering.

6. Discussion

SCALPEL improves segmentation of difficult objects. We analyze the difference in average overlap between SCALPEL and CPMC in Figure 7 when objects are grouped into scale by quintile, where scale is defined as the percentage of image occupied by the object. For tiny ($\approx 10\%$ of total width) and small ($\approx 20\%$ total width) objects, SCALPEL offers a large improvement over CPMC; the total relative improvement for these difficult and small objects is over 100%. SCALPEL performs slightly worse than CPMC for the largest objects, most likely due to the greedy inference being unable to handle occlusions that separate objects into multiple disconnected regions.

Cascades use different features at different stages. We show the weight values for two different features of the cascade in Figure 8. As desired, the cascade learns to weight features differently at different stages of inference; for ex-

Figure 9. Selected segmentation results. Right column: Failure cases. Remaining: Several successful segmentations; note that simply thresholding the prior would result in failures on the more difficult examples.

ample, the *(Large,Person)* cascade learns to down-weight exterior edges until nearing completion of the inference process.

7. Conclusion

We have presented SCALPEL, a novel method for state-of-the-art segment proposal generation with efficient training of class- and scale-specific segmentation cascades. The segments proposed by SCALPEL are more accurate than state-of-the-art competitors as measured by three different error metrics. Furthermore, our approach can be extended to incorporate arbitrary new features or bounding box proposals, and additional specifities besides class and scale (such as shape or color) could be explored as well.

Acknowledgments

The authors were partially supported by a NSF GRFP Fellowship, ONR MURI N000141010934, NSF CAREER 1054215 and by STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

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

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 2011.

[3] E. Borenstein and S. Ullman. Learning to segment. In *ECCV*, 2004.

[4] T. Brox, L. Bourdev, S. Maji, and J. Malik. Object segmentation by alignment of poselet activations to image contours. In *CVPR*, 2011.

[5] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010.

[6] Q. Dai and D. Hoeim. Learning to localize detected objects. In *CVPR*, 2012.

[7] I. Endres and D. Hoeim. Category independent object proposals. In *ECCV*, 2010.

[8] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A libray for large scale linear classification. *JMLR*, 2008.

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2009.

[10] C. Gu, P. Arbelaez, Y. Lin, K. Yu, and J. Malik. Multi-component models for object detection. In *ECCV*, 2012.

[11] A. Ion, J. Carreira, and C. Sminchisescu. Image segmentation by figure-ground composition into maximal cliques. In *ICCV*, 2011.

[12] J. Kim and K. Grauman. Shape sharing for object segmentation. In *ECCV*, 2012.

[13] D. Kuettel and V. Ferrari. Figure-ground segmentation by transferring window masks. In *CVPR*, 2012.

[14] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[15] O. Parkhi, A. Vedaldi, C. Jawahar, and A. Zisserman. The truth about cats and dogs. In *ICCV*, 2011.

[16] E. Rahtu, J. Kannala, and M. Blaschko. Learning a category independent object detection cascade. In *ICCV*, 2011.

[17] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, 2003.

[18] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *CVPR*, 2013.

[19] A. Rosenfeld and D. Weinshall. Extracting foreground masks towards object recognition. In *ICCV*, 2011.

[20] E. van de Sande, J. Uijlingsy, T. Gevers, and A. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011.

[21] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[22] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object models for image segmentation. *PAMI*, 2012.