Multiscale Combinatorial Grouping

Pablo Arbeláez[1,*] Jordi Pont-Tuset[2,*] Jonathan T. Barron[1] Ferran Marques[2] Jitendra Malik[1]

[1]University of California, Berkeley
Berkeley, CA 94720
{arbelaez,barron,malik}@eecs.berkeley.edu

[2]Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
{jordi.pont,ferran.marques}@upc.edu

Abstract

We propose a unified approach for bottom-up hierarchical image segmentation and object candidate generation for recognition, called Multiscale Combinatorial Grouping (MCG). For this purpose, we first develop a fast normalized cuts algorithm. We then propose a high-performance hierarchical segmenter that makes effective use of multiscale information. Finally, we propose a grouping strategy that combines our multiscale regions into highly-accurate object candidates by exploring efficiently their combinatorial space. We conduct extensive experiments on both the BSDS500 and on the PASCAL 2012 segmentation datasets, showing that MCG produces state-of-the-art contours, hierarchical regions and object candidates.

1. Introduction

Two paradigms have shaped the field of object recognition in the last decade. The first one, popularized by the Viola-Jones face detection algorithm [27], formulates object localization as window classification. The basic scanning-window architecture, relying on histograms of gradients and linear support vector machines, was introduced by Dalal and Triggs [7] in the context of pedestrian detection and is still at the core of leading object detectors on the PASCAL challenge such as Deformable Part Models [11].

The second paradigm relies on perceptual grouping to provide a limited number of high-quality and category-independent object candidates, which can then be described with richer representations and used as input to more sophisticated learning methods. Examples in this family are [18, 13]. Recently, this approach has dominated the PASCAL segmentation challenge [6, 3, 5], improved object detection [24] and proven competitive in large-scale classification [26].

*First two authors contributed equally

Figure 1. **Top:** original image, instance-level groundtruth from PASCAL and our multiscale hierarchical segmentation. **Bottom:** our best object candidates among 400.

Since the power of this second paradigm is critically dependent on the accuracy and the number of object candidates, an increasing body of research has delved into the problem of their generation [6, 10, 1, 15]. However, those approaches typically focus on learning generic properties of objects from a set of examples, while reasoning on a fixed set of regions and contours produced by external bottom-up segmenters such as [4, 12].

In this paper, we propose a unified approach to multiscale hierarchical segmentation and object candidate generation called Multiscale Combinatorial Grouping (MCG). Fig. 1 shows an example of our results and Fig. 2 an overview of our pipeline. Our main contributions are:

- An efficient normalized cuts algorithm, which in practice provides a 20× speed-up to the eigenvector computation required for contour globalization [4, 21] (Sect. 3.1).
- A state-of-the-art hierarchical segmenter that leverages multiscale information (Sect. 4).
- A grouping algorithm that produces accurate object candidates by efficiently exploring the combinatorial space of our multiscale regions (Sect. 6).

We conduct a comprehensive empirical validation. On the BSDS500 (Sect. 5) we report the best results to date in contour detection and hierarchical segmentation. On the

1063-6919/14 $31.00 © 2014 IEEE
DOI 10.1109/CVPR.2014.49

VOC2012 segmentation dataset (Sect. 7), our candidates obtain overall state-of-the-art object-level accuracy. At a regime of 1100 candidates per image (c/i), we report the best results on $12/20$ object categories and a relative improvement of $+20\%$ over Selective Search [26]. At 100 c/i, our candidates provide a relative improvement of $+7.8\%$ over CPMC [6].

2. Related Work

For space reasons, we focus our review on recent normalized cut algorithms and object candidates for recognition.

Fast normalized cuts The efficient computation of normalized-cuts eigenvectors has been the subject of recent work, as it is often the computational bottleneck in grouping algorithms. Taylor [25] presented a technique for using a simple watershed oversegmentation to reduce the size of the eigenvector problem, sacrificing accuracy for speed. We take a similar approach of solving the eigenvector problem in a reduced space, though we use simple image-pyramid operations on the affinity matrix (instead of a separate segmentation algorithm) and we see no loss in performance despite a $20\times$ speed improvement. Maire and Yu [16] presented a novel multigrid solver for producing eigenvectors at multiple scales, which speeds up fine-scale eigenvector computation by leveraging coarse-scale solutions. Our technique also uses the scale-space structure of an image, but instead of solving the problem at multiple scales, we simply reduce the scale of the problem, solve it at a reduced scale, and then upsample the solution while preserving the structure of the image. As such, our technique is faster and much simpler, requiring only a few lines of code wrapped around a standard sparse eigensolver.

Object Candidates Class-independent methods that generate object hypotheses can be divided into those whose output is an image window and those that generate segmented candidates.

Among the former, Alexe *et al*. [1] propose an *objectness* measure to score randomly-sampled image windows based on low-level features computed on the superpixels of [12]. Van de Sande *et al*. [26] present a selective window search based on segmentation. Starting with the superpixels of [12] for a variety of color spaces, they produce a set of segmentation hierarchies by region merging, which are used to produce a set of object candidate windows. While we also take advantage of different hierarchies to gain diversity, we leverage multiscale information rather than different color spaces. Furthermore, in contrast to [1, 26] we focus on the finer-grained task of pixel-accurate object extraction, rather than on window selection.

Among the methods that produce segmented candidates, Carreira and Sminchisescu [6] hypothesize a set of placements of fore- and background seeds and, for each configuration, solve a constrained parametric min-cut (CPMC) problem to generate a pool of object hypotheses. Endres and Hoiem [10] base their category-independent object proposals on an iterative generation of a hierarchy of regions, based on the contour detector of [4] and occlusion boundaries of [14]. Kim and Grauman [15] propose to match parts of the shape of exemplar objects, regardless of their class, to detected contours by [4]. They infer the presence and shape of a candidate object by adapting the matched object to the computed superpixels.

Recently, two works proposed to train a cascade of classifiers to learn which sets of regions should be merged to form objects. Ren and Shankhnarovich [22] produce full region hierarchies by iteratively merging pairs of regions and adapting the classifiers to different scales. Weiss and Taskar [28] specialize the classifiers also to size and class of the annotated instances to produce object candidates.

Malisiewicz and Efros [18] took one of the first steps towards combinatorial grouping, by running multiple segmenters with different parameters and merging up to three adjacent regions. In [3], another step was taken by considering hierarchical segmentations at three different scales and combining pairs and triplets of adjacent regions from the two coarser scales to produce object candidates.

A substantial difference between our approach and previous work is that, instead of relying on pre-computed hierarchies or superpixels, we propose a unified approach that produces and groups high-quality multiscale regions. With respect to the combinatorial approaches of [18, 3], our main contribution is to develop efficient algorithms to explore a much larger combinatorial space by taking into account a set of object examples, increasing the likelihood of having complete objects in the pool of candidates. Our approach has therefore the flexibility to adapt to specific applications and types of objects, and can produce candidates at any trade-off between their number and their accuracy.

3. The Segmentation Algorithm

Consider a segmentation of the image into regions that partition its domain $\mathcal{S} = \{S_i\}_i$. A segmentation hierarchy is a family of partitions $\{\mathcal{S}^*, \mathcal{S}^1, ..., \mathcal{S}^L\}$ such that: (1) $\mathcal{S}^*$ is the finest set of *superpixels*, (2) $\mathcal{S}^L$ is the complete domain, and (3) regions from coarse levels are unions of regions from fine levels. A hierarchy where each level $\mathcal{S}^i$ is assigned a real-valued index λ_i can be represented by a dendrogram, a region tree where the height of each node is its index. Furthermore, it can also be represented as an ultrametric contour map (UCM), an image obtained by weighting the boundary of each pair of adjacent regions in the hierarchy by the index at which they are merged [2]. This representation unifies the problems of contour detection and hierarchical image segmentation: a threshold at level λ_i in the UCM produces the segmentation $\mathcal{S}^i$.

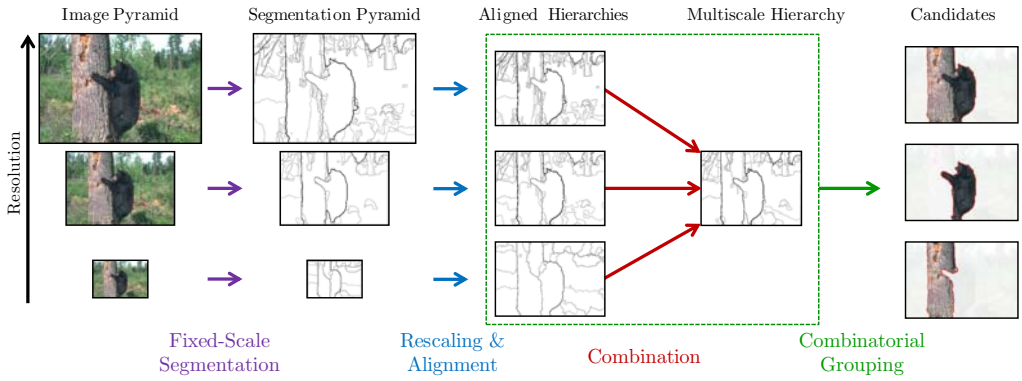

Figure 2. **Multiscale Combinatorial Grouping**. Starting from a multiresolution image pyramid, we perform hierarchical segmentation at each scale independently. We align these multiple hierarchies and combine them into a single multiscale segmentation hierarchy. Our grouping component then produces a ranked list of object candidates by efficiently exploring the combinatorial space of these regions.

As an example, in the gPb-ucm algorithm of [4], brightness, color and texture gradients at three fixed disk sizes are first computed. These local contour cues are globalized using spectral graph-partitioning, resulting in the gPb contour detector. Hierarchical segmentation is then performed by iteratively merging adjacent regions based on the average gPb strength on their common boundary. This algorithm produces therefore a tree of regions at multiple levels of homogeneity in brightness, color and texture, and the boundary strength of its UCM can be interpreted as a measure of contrast.

Coarse-to-fine is a powerful processing strategy in computer vision. We exploit it in two different ways to develop an efficient, scalable and high-performance segmentation algorithm: (1) To speed-up spectral graph partitioning and (2) To create aligned segmentation hierarchies.

3.1. Fast Downsampled Eigenvector Computation

The normalized cuts criterion is a key globalization mechanism of recent high-performance contour detectors such as [4, 21]; Although powerful, such spectral graph partitioning has a significant computational cost and memory footprint that limit its scalability. In this section, we present an efficient normalized cuts algorithm which in practice preserves full performance for contour detection, has low memory requirements and provides a $20\times$ speed-up.

Given a symmetric affinity matrix A, we would like to compute the k smallest eigenvectors of the Laplacian of A. Directly computing such eigenvectors can be very costly even with sophisticated solvers, due to the large size of A. We therefore present a technique for approximating them much more efficiently by taking advantage of the multiscale nature of our problem: A models affinities between pixels in an image, and images naturally lend themselves to multiscale or pyramid-like representations and algorithms.

Our algorithm is inspired by two observations: 1) if A is bistochastic (the rows and columns of A sum to 1) then the eigenvectors of the Laplacian A are equal to the eigenvectors of the Laplacian of A^2, and 2) because of the scale-similar nature of images, the eigenvectors of a "downsampled" version of A in which every other pixel has been removed should be similar to the eigenvectors of A. Let us define $\texttt{pixel\_decimate}(A)$, which takes an affinity matrix A and returns the indices of rows/columns in A corresponding to a decimated version of the image from which A was constructed. That is, if $i = \texttt{pixel\_decimate}(A)$, then $A[i,i]$ is a decimated matrix in which alternating rows and columns *of the image* have been removed. Computing the eigenvectors of $A[i,i]$ works poorly, as decimation disconnects pixels in the affinity matrix, but the eigenvectors of the decimated squared affinity matrix $A^2[i,i]$ are similar to those of A, because by squaring the matrix before decimation we intuitively allow each pixel to propagate information to all of its neighbors in the graph, maintaining connections even after decimation. Our algorithm works by efficiently computing $A^2[i,i]$ as $A[:,i]^{\mathrm{T}} A[:,i]$ (the naive approach of first squaring A and then decimating it is intractable), computing the eigenvectors of $A^2[i,i]$, and then "upsampling" those eigenvectors back to the space of the original image by multiplying by $A[:,i]$. This squaring-and-decimation procedure can be applied recursively several times, improving efficiency while sacrificing accuracy.

Pseudocode for our algorithm, which we call "DNCuts" (downsampled normalized cuts) is given in Alg. 1, where A is our affinity matrix and D is the number of times that our squaring-and-decimation operation is applied. Our algorithm repeatedly applies our joint squaring-and-decimation procedure, computes the smallest k eigenvectors of the final "downsampled" matrix A_D by using a standard sparse eigensolver $\texttt{ncuts}(\texttt{A}_\texttt{D}, \texttt{K})$, and repeatedly "upsamples" those eigenvectors. Because our A is not bistochastic and decimation is not an orthonormal operation, we must do some normalization throughout the algorithm (line 5) and whiten the resulting eigenvectors (line 10). We found that

Algorithm 1 $\texttt{dncuts}(A, D, K)$

$A_0 \leftarrow A$
for $d = [1 : D]$ **do**
  $i_d \leftarrow \texttt{pixel\_decimate}\,(A_{d-1})$
  $B_d \leftarrow A_{d-1}\,[:, i_d\,]$
  $C_d \leftarrow \texttt{diag}(B_d\vec{1})^{-1}B_d$
  $A_d \leftarrow C_d^{\mathrm{T}}B_d$
$X_D \leftarrow \texttt{ncuts}(A_D, K)$
for $d = [D : -1 : 1]$ **do**
  $X_{d-1} \leftarrow C_d X_d$
return $\texttt{whiten}(X_0)$

values of $K = 2$ or $K = 3$ worked well in practice. Larger values of K yielded little speed improvement (as much of the cost is spent downsampling A_0) and start hurting performance. Our technique is similar to Nystrom's method for computing the eigenvectors of a subset of A, but our squaring-and-decimation procedure means that we do not depend on long-range connections between pixels.

3.2. Aligning Segmentation Hierarchies

Spatially transforming an UCM is nontrivial because its boundaries are one-dimensional entities whose topology and strength determine the underlying hierarchy, and an error at a single pixel can have drastic effects. We therefore opt for sampling uniformly K levels in the hierarchy, transforming them sequentially, and reconstructing from their boundaries a transformed UCM.

We consider two different segmentations $\mathcal{R} = \{R_i\}_i$ and $\mathcal{S} = \{S_j\}_j$. We define the projection of the segmentation $\mathcal{R}$ onto a region $S_j \in \mathcal{S}$ as the majority label

$$\pi(\mathcal{R}, S_j) = \operatorname*{argmax}_i \frac{|S_j \cap R_i|}{|S_j|} \tag{1}$$

And the projection of $\mathcal{R}$ onto $\mathcal{S}$ as

$$\pi(\mathcal{R}, \mathcal{S}) = \{\pi(\mathcal{R}, S_j)\}_j. \tag{2}$$

In order to project an UCM onto a target segmentation $\mathcal{S}$, which we denote $\pi(\text{UCM}, \mathcal{S})$, we project each of the levels in the hierarchy in turn.

In the next section, we will iterate this procedure, and project an UCM recursively to a set of target segmentations $\{\mathcal{S}^{1*}, ..., \mathcal{S}^{N*}\}$. However, note that the composition of two such projections can be written as :

$$\pi(\pi(\text{UCM}, \mathcal{S}^1), \mathcal{S}^2) = \pi(\text{UCM}, \mathcal{S}^1) \circ \pi(\mathcal{S}^1, \mathcal{S}^2). \tag{3}$$

In practice, this property means that successive projections of the target segmentations can be pre-computed, the UCM has to be projected only to the first target segmentation, and its final labels are obtained by $N - 1$ look-ups. This procedure is summarized in pseudo-code in Algorithm 2[1].

[1]Note that, by construction, the routines `sampleHierarchy` and `extractBoundary` are fast, as they involve only connected component labeling and thresholding operations. The complexity is thus dominated by the transformations in Steps 6 and 7, which are computed K times.

Algorithm 2 UCM Rescaling and Alignment

Require: An UCM and a set of levels $[t_1, ..., t_K]$
Require: A set of target segmentations $\{\mathcal{S}^{1*}, ..., \mathcal{S}^{N*}\}$
Pre-compute target projections:
$\pi(\mathcal{S}^{1*}, \mathcal{S}^{2*})$, $\pi(\pi(\mathcal{S}^{1*}, \mathcal{S}^{2*}), \mathcal{S}^{3*})$, ...
$\text{UCM}_\pi \leftarrow 0$
for $t = [t_1, ..., t_K]$ **do**
  $\mathcal{S} \leftarrow \texttt{sampleHierarchy}(\text{UCM}, t)$
  $\mathcal{S} \leftarrow \texttt{rescaleSegmentation}(\mathcal{S}, \mathcal{S}^{1*})$
  $\mathcal{S} \leftarrow \pi(\mathcal{S}, \mathcal{S}^{1*})$
  $\mathcal{S} \leftarrow \texttt{readLabels}(\mathcal{S}, \{\mathcal{S}^{1*}, ..., \mathcal{S}^{N*}\})$
  $contours \leftarrow \texttt{extractBoundary}(\mathcal{S})$
  $\text{UCM}_\pi \leftarrow \texttt{max}(\text{UCM}_\pi, t * contours)$
return UCM_π

4. Multiscale Hierarchical Segmentation

Single-scale segmentation We consider as input the following local contour cues: (1) brightness, color and texture differences in half-disks of three sizes [19], (2) sparse coding on patches [21], and (3) structured forest contours [8]. We globalize the contour cues independently using our fast eigenvector gradients of Sect. 3.1, combine global and local cues linearly, and construct an UCM based on the mean contour strength. We tried learning weights using gradient ascent on the F-measure on the train set [4], but evaluating the final hierarchies rather than open contours. We observed that this objective favors the quality of contours at the expense of regions and obtained better overall results by optimizing the Cover metric.

Hierarchy Alignment We construct a multiresolution pyramid with N scales by subsampling / supersampling the original image and applying our single-scale segmenter. In order to preserve thin structures and details, we declare as set of possible boundary locations the finest superpixels $\mathcal{S}^{N*}$ in the highest-resolution. We extract the finest superpixels of each hierarchy, rescale them to the original image resolution, pre-compute their successive projections to $\mathcal{S}^{N*}$ and then transfer recursively the strength of all the coarser UCMs by applying Algorithm 2.

Multiscale Hierarchy After alignment, we have a fixed set of boundary locations, and N strengths for each of them, coming from the different scales. We formulate this as a binary boundary classification problem and train a classifier that combines these N features into a single probability of boundary estimation. We experimented with several learning strategies for combining UCM strengths: (a) Uniform weights transformed into probabilities with Platt's method.

(b) SVMs and logistic regression, with both linear and additive kernels. (c) Random Forests. (d) The same algorithm as for single-scale. We found the results with all learning methods surprisingly similar, in agreement with the observation reported by [19]. This particular learning problem, with only a handful of dimensions and millions of data points, is relatively easy and performance is mainly driven by our already high-performing and well calibrated features. We therefore use the simplest option (a).

5. Experiments on the BSDS500

We conduct extensive experiments on the BSDS500 [23], using the standard evaluation metrics and following the best practice rules of that dataset. We also report results with a recent evaluation metric F_{op} [20], Precision-Recall for objects and parts, using the publicly-available code.

Single-scale Segmentation Table 1-top shows the performance of our single-scale segmenter for different types of input contours on the validation set of the BSDS500. We obtain high-quality hierarchies for all the cues considered, showing the generality of our approach. Furthermore, when using them jointly (row 'single-combined'), our segmenter outperforms the versions with individual cues, suggesting its ability to leverage diversified inputs. In terms of efficiency, our fast normalized cuts algorithm provides an average 20× speed-up over [4], starting from the same local cues, with no significant loss in accuracy and with a low memory footprint.

Multiscale Segmentation Table 1-bottom evaluates our full approach in the same experimental conditions as the upper panel. We observe a consistent improvement in performance in all the metrics for all the inputs, which validates our architecture for multiscale segmentation. We experimented with the range of scales and found $N = \{0.5, 1, 2\}$ adequate for our purposes. A finer sampling or a wider range of scales did not provide noticeable improvements. We tested also two degraded versions of our system (not shown in the table). For the first one, we resized contours to the original image resolution, created UCMs and combined them. For the second one, we transformed per-scale UCMs to the original resolution, but omitted the strength transfer to the finest superpixels. The first ablated version produces interpolation artifacts and smooths away details, while the second one suffers from misalignment. Both fail to improve the performance of the single-scale result, which provides additional empirical support for our multiscale approach.

Since there are no drastic changes in our results when taking as input the different individual cues or their combination, in the sequel we use the version with structured forests for efficiency reasons, which we denote *Ours-multi*.

Comparison with state-of-the-art. Figure 3 compares our multiscale hierarchical segmenter on the BSDS500 test

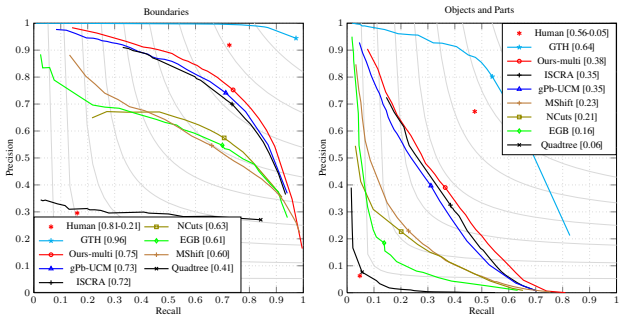

Figure 3. **BSDS500 test set.** Precision-Recall curves for boundaries [23] (left) and for objects and parts [20] (right). The marker on each curve is placed on the Optimal Dataset Scale (ODS). The isolated red asterisks refer to the human performance assessed on the same image and on a swapped image. In the legend, the F measure of the marked point on each curve is presented in brackets.

		Boundary		Region							
		F_b		F_{op}		SC		PRI		VI	
	Input	ODS	OIS	ODS	OIS	ODS	OIS	ODS	OIS	ODS	OIS
Single-Scale	Pb [19]	0.702	0.733	0.334	0.370	0.577	0.636	0.801	0.847	1.692	1.490
	SC [21]	0.697	0.725	0.264	0.306	0.540	0.607	0.777	0.835	1.824	1.659
	SF [8]	0.719	0.737	0.338	0.399	0.582	0.651	0.803	0.851	1.608	1.432
	Comb.	0.719	0.750	0.358	0.403	0.602	0.655	0.809	0.855	1.619	1.405
Multiscale	Pb [19]	0.713	0.745	0.350	0.389	0.598	0.656	0.807	0.856	1.601	1.418
	SC [21]	0.705	0.734	0.331	0.384	0.579	0.647	0.799	0.851	1.637	1.460
	SF [8]	0.725	0.744	0.370	**0.420**	0.600	0.660	0.810	0.854	1.557	1.390
	Comb.	**0.725**	**0.757**	**0.371**	0.408	**0.611**	**0.670**	**0.813**	**0.862**	**1.548**	**1.367**

Table 1. **BSDS500 val set.** Control experiments for single-scale (top) and multiscale (bottom) hierarchical segmentation.

set against all the methods for which there is publicly available code. We also compare to the recent ISCRA [22] hierarchies, provided precomputed by the authors. We obtain consistently the best results to date on the BSDS for all operating regimes, both in terms of boundary and region quality.

Note that the range of object scales in the BSDS500 is limited, which translates into modest absolute gains for multiscale segmentation. However, we will observe more substantial improvements when we move to PASCAL in Section 7 (See Fig. 5).

Ground-Truth Hierarchy. In order to gain further insights, we transfer the strength of each ground-truth segmentation to our highest-resolution superpixels $\mathcal{S}^{N*}$ and construct a combined hierarchy. This approximation to the semantic hierarchy, "GTH" in Fig. 3, is an upper-bound for our approach as both share the same boundary locations and the only difference is their strength. Since the strength of GTH is proportional to the number of subjects marking it, it provides naturally the correct semantic ordering, where outer object boundaries are stronger than internal parts.

Recently, Maire *et al*. [17] developed an annotation tool where the user encodes explicitly the "perceptual strength" of each contour. Our approach provides an alternative where the semantic hierarchy is reconstructed by sampling flat annotations from multiple subjects.

6. Object Candidate Generation

Our proposal for object candidate generation is to create a large-enough set of hypotheses with a very high achievable quality (Sect. 6.1) and then to learn to rank them using low-level features (Sect. 6.2) to keep the maximum quality.

6.1. Combinatorial Grouping of Candidates

We consider the singletons, pairs, triplets, and 4-tuples of regions from the three individual scales and the multiscale hierarchy as 16 lists of ranked candidates. Since the full joint set of candidates is very large and redundant, our goal is to reduce it by keeping only the top N_i candidates from each ranked list L_i, resulting in a total number of candidates $N_c = \sum N_i$.

At training time, we would like to find, for different values of N_c, the number of candidates from each list $\overline{N}_i$ such that the joint pool of N_c candidates has the best achievable quality. We frame this learning problem as a Pareto front optimization [9] with two conflicting objective functions: number of candidates and achievable quality. At test time, we select a working point on the Pareto front, represented by the $\{\overline{N}_i\}$ values, based either on the number of candidates N_c we can handle or on the minimum achievable quality our application needs, and we combine the $\overline{N}_i$ top candidates from each ranked list.

Efficient learning: Formally, assuming R ranked lists L_i, an exhaustive learning algorithm would consider all possible values of the R-tuple $\{N_1, \ldots, N_R\}$, where $N_i \in \{0, \ldots, |L_i|\}$; adding up to $\prod_1^R |L_i|$ parameterizations to try, which is intractable in our setting.

To reduce the dimensionality of the learning step, we start by selecting two ranked lists L_1, L_2 and we sample the list at S levels of number of candidates. We then scan the full S^2 different parameterizations to combine the candidates from both. Each of these sets is analyzed in the plane of number of candidates-achievable quality, so the full combination can be assessed as S^2 points in this plane.

The key step of the optimization consists in discarding those parameterizations whose quality point is not in the Pareto front. We sample the Pareto front to S points and we iterate the process until all the ranked lists are combined. Each point in the final Pareto front corresponds to a particular parameterization $\{N_1, \ldots, N_R\}$. At test time, we choose a point on this curve, either at a given number of candidates N_c or at the achievable quality we are interested in, and combine the $\{\overline{N}_1, \ldots, \overline{N}_R\}$ top candidates from each ranked list. The number of sampled configurations using the proposed algorithm is $(R-1)S^2$, that is, we have reduced an exponential problem (S^R) to a quadratic one.

6.2. Regressed Ranking of Candidates

The previous section tackles the reduction of candidates from millions to thousands while keeping the achievable quality as high as possible. To further reduce their number, we train a regressor from low-level features, as in [6].

We focus on features that can be computed efficiently in a bottom-up fashion. This way, we can precompute all the features for the original regions and efficiently calculate the features for the candidates in a few operations. We compute the following features:

- **Size and location**: Area and perimeter of the candidate; area, position, and aspect ratio of the bounding box; and the area balance between the regions in the candidate.
- **Shape**: Perimeter (and sum of contour strength) divided by the squared root of the area; and area of the region divided by that of the bounding box.
- **Contours**: Sum of contour strength at the boundaries, mean contour strength at the boundaries; minimum and maximum UCM threshold of appearance and disappearance of the regions forming the candidate.

We train a Random Forest using these features to regress the object overlap with the ground truth, and diversify the ranking based on Maximum Marginal Relevance measures [6].

7. Experiments on PASCAL 2012

Evaluation Measures: We assess the generation of candidates in terms of achievable quality with respect to the number of candidates, that is, the quality we would have if an oracle selected the best candidate among the pool. As a measure of quality of a specific candidate with respect to an annotated object, we will consider the Jaccard index, also known as covering, or overlap, which is defined as the intersection over the union of two sets.

When computing the overall quality for the whole database, we will consider two metrics. First, we define the Jaccard index at class level (J_c) as the mean over classes of the covering of all pixels of each class (the segmentation accuracy of PASCAL). Second, to avoid the bias of J_c towards methods focusing on large objects, we define the Jaccard index at instance level (J_i) as the mean best overlap for all the ground-truth instances in the database (also Best Spatial Support score (BSS) [18]).

Learning Strategy Evaluation: This section estimates the loss in performance due to not sweeping all the possible values of $\{N_1, \ldots, N_R\}$ via the greedy strategy proposed. To do so, we will compare it with the full combination on a reduced problem to make the latter feasible. Specifically, we combine the 4 ranked lists coming from the singletons at all scales, instead of the full 16 lists, and we limit the search to 20 000 candidates.

In this situation, the mean loss in achievable quality along the full curve of parameterization is $J_i = 0.0002$, with

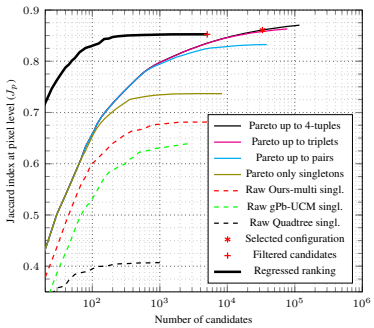

Figure 4. **VOC12 train set.** Object candidates achievable quality

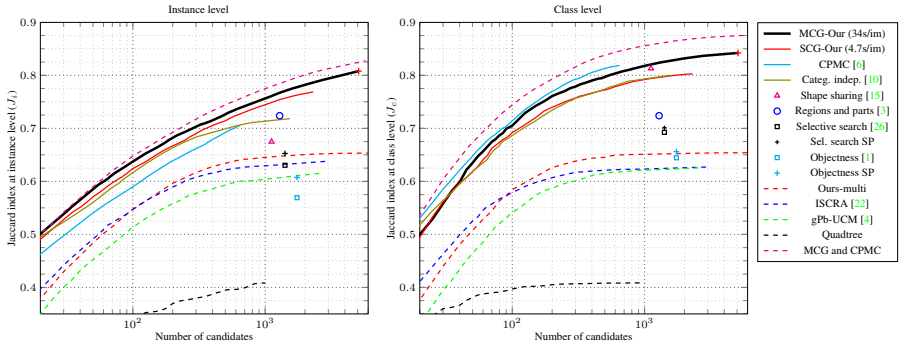

Figure 5. **VOC12 val set.** Object candidates achievable quality at instance and class level. We also compare favorably to Scalpel [28] on VOC10 val set.

	N_c	Plane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	MBike	Person	Plant	Sheep	Sofa	Train	TV	Global
MCG	5086	82.9	52.0	85.6	80.5	77.2	81.8	77.7	90.5	74.8	88.1	84.5	87.6	82.8	78.8	79.7	77.4	87.1	87.2	84.0	89.6	80.8
MCG	1100	**80.0**	47.8	**83.9**	**76.4**	**71.1**	78.5	**68.9**	89.3	**68.5**	**85.9**	79.8	85.8	**80.4**	75.4	**73.5**	**69.3**	**84.9**	82.6	81.7	**85.8**	**76.0**
[10]	1100	75.1	**49.1**	80.7	68.8	62.8	76.4	63.3	**89.4**	64.6	83.0	80.3	83.7	78.4	**78.0**	66.9	66.2	69.5	82.0	**84.3**	81.8	71.6
[3]	1100	74.4	46.6	80.5	69.4	64.6	73.5	61.2	89.0	65.1	80.5	78.4	85.2	77.2	70.6	67.9	68.8	73.5	81.6	75.8	82.0	71.4
[15]	1100	73.8	40.6	75.8	66.7	52.7	**79.7**	50.6	91.2	59.2	80.2	**80.7**	**87.4**	79.0	74.7	62.1	54.6	65.0	**84.6**	82.4	79.5	67.4
[26] ad.	1100	68.3	39.6	70.6	64.8	58.0	68.2	51.8	77.6	58.2	72.6	70.4	74.0	66.2	59.9	59.8	55.4	67.7	71.3	68.6	78.7	63.1
[26]	1100	55.9	34.6	56.0	62.0	55.1	76.2	50.4	70.7	52.4	67.7	65.0	67.4	58.2	59.9	55.2	53.5	59.5	68.5	70.8	85.7	58.9
MCG	100	70.2	38.8	73.6	**67.7**	**55.3**	**68.5**	**50.6**	82.4	54.4	**78.1**	67.7	77.7	69.3	66.3	**59.9**	51.4	**70.2**	**74.1**	72.6	**78.1**	**63.7**
[10]	100	70.6	**40.8**	**74.8**	59.9	49.6	65.4	50.4	81.5	**54.5**	74.9	**68.1**	77.3	69.3	**66.8**	56.2	**54.3**	64.1	72.0	71.6	69.9	61.7
[6]	100	**72.7**	36.2	73.6	63.3	45.4	67.4	39.5	**84.1**	47.7	73.2	64.0	**81.1**	**72.2**	64.3	52.8	42.9	62.2	72.9	**74.3**	69.5	59.0

Table 2. **VOC12 val set.** Per-class and global Jaccard index at instance level (J_i)

a maximum loss of $J_i = 0.004$ (0.74%). In exchange, our proposed learning strategy on the full 16 ranked lists takes about 1 minute to compute on the training set of PASCAL 2012, while the limited full combination takes 4 days

Combinatorial Grouping: We extract the lists of candidates from the three scales and the multiscale hierarchy, for singletons, pairs, triplets, and 4-tuples of regions, leading to 16 lists, ranked by the minimum UCM strength of the regions forming each candidate.

Figure 4 shows the Pareto front evolution of J_c with respect to the number of candidates for up to 1, 2, 3, and 4 regions per candidate (4, 8, 12, and 16 lists, respectively) at training time. As baselines, we plot the raw singletons from Ours-multi, gPb-UCM, and Quadtree.

The improvement of considering the combination of all 1-region candidates from the 3 scales and the Ours-multi with respect to the raw Ours-multi is significant, which corroborates the diversity obtained from hierarchies at different scales. In turn, the addition of 2- and 3-region candidates noticeably improves the achievable quality. The improvement when adding 4-tuples is marginal at the number of candidates we are working.

The red asterisk (*) marks the selected configuration $\{\overline{N}_1, \ldots, \overline{N}_R\}$ we choose, and the red plus sign (+) represents the set of candidates after removing duplicates with an overlap higher than 0.95. The candidates at this point (5 038 per image in mean with $J_c = 0.85$) are the ones that are ranked by the learnt regressor, the result of which is plotted in black (on the training set).

In summary, the full set of candidates (i.e., combining the full 16 lists) would contain millions of candidates per image. In the validation set of PASCAL 2012, the multiscale combinatorial grouping allows us to reduce the number of candidates to 5 086 with a very high achievable J_c of 0.84. The regressed ranking allows us to further reduce the number of candidates below this point.

Comparison with State of the Art: We compare our results against [1, 15, 6, 3, 26, 10], using the implementations from the respective authors.

Figure 5 shows the achievable quality of all methods on the validation set of PASCAL 2012. We plot the raw regions of Ours-multi, ISCRA, gPb-UCM, and QuadTree as baselines. To compare more fairly with selective search [26] and objectness [1], we adapted their boxes to the superpixels of our multiscale hierarchy, obtaining a small improvement.

At instance level (J_i), MCG candidates outperform the state-of-the-art at all regimes. At class level (J_c), our candidates practically achieve the same quality as CPMC. To evaluate their complementarity, we compute the Pareto front of combining the two sets of ranked candidates; that is, we evaluate the sets of candidates corresponding to combining some candidates from MCG and CPMC. The curve obtained (dashed magenta - - -), shows that MCG and CPMC are very complementary: the combination of both methods leads to an improvement of $J_c = 0.03$ at around 650 c/i.

We also present a faster single-scale version of MCG (*SCG*), which takes the hierarchy at the native scale only and combines up to three regions. We decrease the timing

one order of magnitude while keeping competitive results.

Table 2 shows the quality (J_i) on each of the 20 PASCAL classes at two different number of candidates (100 and 1100), comparing MCG with the relevant state-of-the-art techniques at that number of candidates. MCG outperforms all techniques on the two regimes at the global level and the majority of classes.

MCG and SCG Timing: Table 3 shows the timings of our approach from scratch, on a single core using less than 2Gb of RAM. Our full MCG takes about 25 s. per image to compute the multiscale hierarchies, and 10 s. to generate and rank 5 038 candidates on the validation set of PASCAL 2012. Our single-scale version produces a segmentation hierarchy of quality comparable to gPb-ucm [4] in just 3 s.

	Hierarchical Segmentation	Candidate Generation	Total
MCG	24.4 ± 3.6	9.9 ± 3.5	34.3 ± 6.2
SCG	3.2 ± 0.4	1.5 ± 0.5	4.7 ± 0.7

Table 3. Time in seconds per image of MCG and SCG

8. Conclusions

We proposed Multiscale Combinatorial Grouping, a unified approach for bottom-up segmentation and object candidate generation. Our approach produces state-of-the-art contours, hierarchical regions and object candidates. At its core are a fast eigenvector computation for normalized-cut segmentation and an efficient algorithm for combinatorial merging of hierarchical regions. In order to promote reproducible research on perceptual grouping, all the resources of this project – code, results and evaluation protocols – are publicly available.

Acknowledgements This work was partially supported by the ONR MURI N00014-10-10933, and the Spanish *Ministerio de Ciencia e Innovación*, under project TEC2010-18094 and FPU grant AP2008-01164.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 34:2189–2202, 2012. 1, 2, 7

[2] P. Arbeláez. Boundary extraction in natural images using ultrametric contour maps. In *POCV*, June 2006. 2

[3] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *CVPR*, 2012. 1, 2, 7

[4] P. Arbeláez, M. Maire, C. C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 1, 2, 3, 4, 5, 7, 8

[5] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, pages VII: 430–443, 2012. 1

[6] J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *PAMI*, 34(7):1312–1328, 2012. 1, 2, 6, 7

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1

[8] P. Dollár and C. Zitnick. Structured forests for fast edge detection. *ICCV*, 2013. 4, 5

[9] M. Ehrgott. *Multicriteria optimization*. Springer, 2005. 6

[10] I. Endres and D. Hoiem. Category-independent object proposals with diverse ranking. *PAMI*, 36(2):222–234, 2014. 1, 2, 7

[11] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9), 2010. 1

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59:2004, 2004. 1, 2

[13] C. Gu, J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *CVPR*, 2009. 1

[14] D. Hoiem, A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *IJCV*, 91(3):328–346, 2011. 2

[15] J. Kim and K. Grauman. Shape sharing for object segmentation. In *ECCV*, 2012. 1, 2, 7

[16] M. Maire and S. X. Yu. Progressive multigrid eigensolvers for multiscale spectral segmentation. *ICCV*, 2013. 2

[17] M. Maire, S. X. Yu, and P. Perona. Hierarchical scene annotation. In *BMVC*, 2013. 5

[18] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *BMVC*, 2007. 1, 2, 6

[19] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *PAMI*, 26(5):530–549, 2004. 4, 5

[20] J. Pont-Tuset and F. Marques. Measures and meta-measures for the supervised evaluation of image segmentation. In *CVPR*, 2013. 5

[21] X. Ren and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 1, 3, 4, 5

[22] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *CVPR*, 2013. 2, 5, 7

[23] B. S. Resources. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html. 5

[24] A. Y. S. Fidler, R. Mottaghi and R. Urtasun. Bottom-up segmentation for top-down detection. *CVPR*, 2013. 1

[25] C. J. Taylor. Towards fast and accurate segmentation. *CVPR*, 2013. 2

[26] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011. 1, 2, 7

[27] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, May 2004. 1

[28] D. Weiss and B. Taskar. Scalpel: Segmentation cascades with localized priors and efficient learning. In *CVPR*, 2013. 2, 7