Scale-Aware Alignment of Hierarchical Image Segmentation

Yuhua Chen Dengxin Dai Jordi Pont-Tuset Luc Van Gool
Computer Vision Lab, ETH Zurich
{yuhua.chen,dai,jponttuset,vangool}@vision.ee.ethz.ch

Abstract

Image segmentation is a key component in many computer vision systems, and it is recovering a prominent spot in the literature as methods improve and overcome their limitations. The outputs of most recent algorithms are in the form of a hierarchical segmentation, which provides segmentation at different scales in a single tree-like structure. Commonly, these hierarchical methods start from some low-level features, and are not aware of the scale information of the different regions in them. As such, one might need to work on many different levels of the hierarchy to find the objects in the scene. This work tries to modify the existing hierarchical algorithm by improving their alignment, that is, by trying to modify the depth of the regions in the tree to better couple depth and scale. To do so, we first train a regressor to predict the scale of regions using mid-level features. We then define the anchor slice as the set of regions that better balance between over-segmentation and under-segmentation. The output of our method is an improved hierarchy, re-aligned by the anchor slice. To demonstrate the power of our method, we perform comprehensive experiments on the BSDS500 dataset, which shows that our method, as a post-processing step, can significantly improve the quality of the hierarchical segmentation representations. We also prove that the improvement generalizes well across different algorithms, with a low computational cost. ***[Yuhua:Add Pascal coco???]***

1. Introduction

Generic image segmentation has been part of computer vision and image processing communities since the advent of these fields many decades ago. The definition of the problem, although vague, is easy to give and understand: "to divide the pixels of an image into different pieces, where each piece represents a distinguished *thing* in the image." Martin *et al*. [21] provided these instructions to annotators to create the Berkeley Segmentation Database (BSDS), which proved that the problem of image segmentation was, indeed, well defined, as humans provided consistent partitions of the images *up to refinement*. In other words, image segmentation is inherently a multi-scale problem.

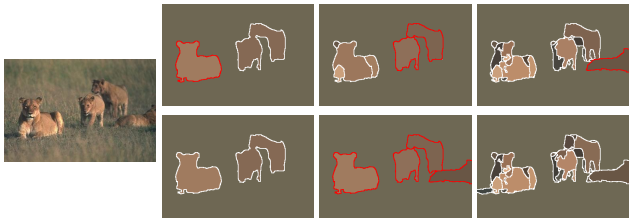

Figure 1. **Example of improved hierarchy alignment**: The original hierarchy (top row) needs three different flat partitions to represent the four objects (highlighted in red). Our aligned hierarchy (bottom row) correctly puts all objects in the same level.

We refer to *flat* image segmentation techniques as those whose output is a single partition of the image pixels into sets [30, 5, 12]. In these cases, in order to capture the aforementioned multi-scale nature of objects, one needs to sweep different parameterizations to obtain multiple partitions that contain the different scales when working with flat segmentation techniques.

On the other hand, hierarchical segmentation produces a single multi-scale structure that aims at capturing the objects at all scales [1, 16, 29, 26, 2]. These types of structures have been successfully used in image filtering [29], semantic segmentation [17], object proposals generation [2], or video segmentation [33, 32].

The representation power of these hierarchies comes at a cost, however, which is the difficulty to handle them from a practical (coding) point of view. While a flat partition can be represented by a matrix of labels of each pixel, hierarchical structures need a much more complex representation. In this context, the Ultrametric Contour Map (UCM) [1] representation is the one that gained more traction and it is widely used in the literature. In it, *flattening* the hierarchy can be achieved by simply *thresholding* the UCM.

The process of *flattening* or *pruning* a hierarchy is therefore of paramount importance for segmentation, because it is the main proxy used towards the final application. This work presents a novel technique to improve the flattening of any given hierarchy, that is, to get better flat partitions from the same hierarchical segmentation.

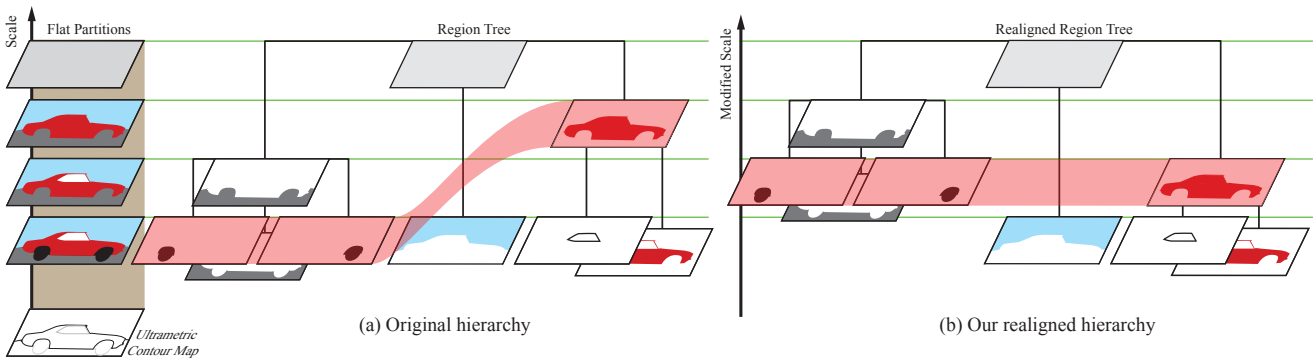

Figure 2. **Our proposed hierarchy realignment**: Given a hierarchy (a) in which the objects at the same scale are not well aligned (represented in the same scale level), we produce a realigned hierarchy (b) that has the similar-scale regions in the same level.

Figure 1 motivates this work. In the first row we can see different flat partitions extracted from the same hierarchy. To get the regions representing the four lions we need to search in three different flat partitions, extracted at three different levels of the hierarchy. The second row shows our results, where the same hierarchy is *aligned* to have all objects represented in the same flat partition.

In other words, the threshold level of the hierarchy better relates with the scale of the objects, not only in the same image, but also across images. To further grasp the intuition of our work, Figure 2 shows a UCM and its interpretation as a region tree (a). In it, the needed regions to form the car are spread into different scale levels (thresholds of the UCM), as marked by the red band. Our proposed realigned hierarchy (b) aims at containing them all in the same scale.

Since the hierarchies are constructed based on low-level features (edges or color), the scale of the objects is not imposed to be coherent. We propose to learn the concept of object scale from mid-level features within the hierarchy. Our objective is to take advantage of these mid-level features as much as possible without getting to high-level features that would allow us to go beyond scale. This way, the global approach would be to construct the hierarchies using low-level features, and then exploit mid-level features to realign them, thus taking the maximum advantage of the most simple features possible.

Our alignment also aims at providing a global alignment among different images, that is, providing levels of scale that keep meaning even when changing images, allowing higher-level methods to generalize in a more straightforward manner. Specifically, we train a regressor to predict whether each region of the hierarchy is oversegmented, undersegmented, or correctly segmented; and we rescale the hierarchy according to the prediction of this classifier. Back to the example in Figure 1, the majority of regions in the first column (bottom) are undersegmented, in the middle column they are correctly segmented, and oversegmented in the last column.

We perform comprehensive experiments on BSDS500 using four different hierarchical segmenters. We obtain consistent improvements on all hierarchies which proves the usefulness of our approach and its generalization power. The remainder of the paper is organized as follows. First, Section 2 gives a brief overview of the related work. Then Section 3 presents our algorithm for re-scaling and aligning hierarchies. We demonstrate the effectiveness of our method in the experiments in Section 4 and draw the conclusions in Section 5.

2. Related Work

Hierarchical Segmentation There is a rich literature of hierarchical segmentation. As stated in the introduction, our focus in this paper is not to develop a better hierarchical segmentation algorithm, but to provide a better alignment of a given hierarchy. Hierarchical segmentation typically starts from various local information embedded in an affinity matrix, such as Pointwise Mutual Information [15], or multiscale local brightness, color, and texture cues [1]. It then greedily constructs a hierarchy of regions by iteratively merging the most similar sets of regions according to a certain metric. The result of hierarchical segmentation is commonly represented as an Ultrametric Contour Map (UCM), where different levels of segmentation can be produced by applying different thresholds to UCM. This work proposes to realign the hierarchies in order to make the thresholds of the UCM more closely related to the scale of objects. Hierarchical segmentation has become the major trend in image segmentation and most of top-performance segmenters [1, 2, 26, 16] fall into this category.

Multiple Segmentations Working with multiple segmentations at the same time has been used in the computer vision community for a long time, with the idea that, while none of the segmentations is likely to partition the image perfectly, some parts in some segmentations might be use-

ful. Hoiem *et al*. [13] use this idea to estimate the scene structure. A similar idea was exploited by Russell *et al*. [28] to discover objects, and by Malisiewicz *et al*. [20] to improve the spatial support of regions for recognition. By re-aligning the hierarchies we aim to minimize the number of partitions from a hierarchy needed to obtain reasonable results, since we concentrate same-scale regions in the same partition. Our work also shares some similarities with [33], where they flatten supervoxel hierarchies in videos by finding a slice with uniform entropy.

Predicting Segmentation Quality by Classification Classification has been exploited to predict segmentation quality in many works. Ren *et al*. [25] use a linear classifier base on Gestalt features [22] to distinguish good and bad segmentations. Their negative training data are generated by randomly placing a ground-truth mask over an image. A similar idea is used to select parameters by Peng *et al*. [23] to select λ in graph-cut based interactive segmentation. They compute the segmentation with different λ, then select the one with highest predicted quality. More recently, Carreira *et al*. [3], Arbelaez *et al*. [2], and Endres *et al*. [9] use a regression forest to predict the good overlap between segments (object proposals) and ground truth objects. We use similar features to [3], which are based on graph partition properties, region properties, and Gestalt properties.

Scale-aware Vision Algorithms Our work also bear a resemblance to the scale-aware algorithms for other vision tasks. For instance, exploiting the scale information has proven helpful for semantic image segmentation [4] and pedestrain detection [18]. [8] show that vision algorithms employing super-resolved images (higher-resolution) perform better than using low-resolution images directly. Other scale-aware applications include object tracking [**?**] and image thumbnailling [31].

3. Flattening and Re-scaling Hierarchies

As discussed in the introduction, while segmentation hierarchies contain a rich multiscale decomposition of the image, it is not trivial to distill such knowledge because the hierarchies generated by current methods are not fully scale-aware. Simply taking a layer yields a segmentation of which some parts are under-segmented while others are over-segmented. In this section, we present our method which aligns the scales of segmentation hierarchies, making image hierarchies easier to use in practice. We start with scale labeling, and then present the alignment strategy.

3.1. Flattening Hierarchies via Scale Labeling

Let's denote the segmentation tree of image I by $\mathcal{T}$, with node v_i indicating its i-th node. The nodes correspond to regions (segments) of I. Given $\mathcal{T}$, our task is to find a tree slice $\mathcal{L}$ to divide all nodes v_i's (segments) into three groups: $\mathcal{L}^-$, $\mathcal{L}$, and $\mathcal{L}^+$ indicating under-, properly- and over-segmented, respectively. See Figure 3(b) for an example of nodes in the three groups.

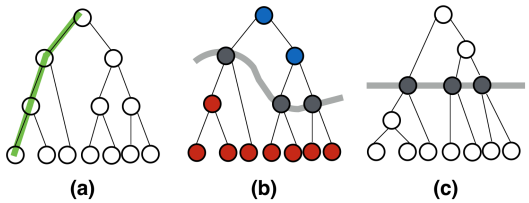

Figure 3. Examples of the slices and paths of the segmentation tree, where one path of the tree is shown in green (a) and one slice is shown in grey (b). In (b), all nodes in blue are in $\mathcal{L}^-$, and all nodes in red are in $\mathcal{L}^+$. Our approach re-aligns the hierarchy using the anchor slice. The aligned tree is shown in (c).

The visual representation of a slice can be seen in Figure 2 as red bands covering different regions. An example of the three types of slices can be found in Figure 1 (bottom row), where the left partition is mainly oversegmented, the middle one correctly segmented, and the right one undersegmented.

The problem is formulated as a three-class labeling problem. For each node v_i, we use $x(v_i) \in \{-1, 0, 1\}$ as its class label, with -1, 0, and 1 indicating the membership of v_i to $\mathcal{L}^-$, $\mathcal{L}$, and $\mathcal{L}^+$ respectively. Assume now that a function $f(v_i) : v_i \rightarrow [-1, 1]$ is provided to measure the granularity of image segments, where negative values stand for under-segmented, 0 for properly-segmented, and positive for over-segmented regions. The magnitude of $f(v_i)$ signals the deviation from being properly-segmented. Section 3.1.2 present the proposed learning algorithm for $f(v_i)$.

The labeling of all v_i's could be done by greedily taking the best-scoring class for each node. However, not any labeling represents a valid slice of the tree. Following the definition in [24, 33], a tree slice is a set of nodes such that every path $\mathcal{P}_n, n \in \{1, 2, ..., N\}$ from the leaf node $\bar{v}_n$ to the root node v_0 contains one and only one node v in the slice. See Figure 3 for the examples of the slices and paths.

From the nature of segmentation hierarchies, the labels of parent nodes v_i^p should be equal or smaller than their child nodes v_i. Intuitively, if a region is correctly segmented, the parent cannot be oversegmented. On the other hand, the parent of an undersegmented region will also be undersegmented. Putting the two constraints together, the labeling problem can be formulated as:

$$\hat{\mathbf{X}} = arg\min_{\mathbf{X}} E(\mathbf{X})$$
$$E(\mathbf{X}) = \sum_{v_i \in \mathcal{L}} \#(v_i) \cdot \|f(v_i)\|^2 + \lambda \sum_{v_i \notin \mathcal{L}} \#(v_i) \cdot l(v_i)$$
$$\text{s.t} \quad \forall n: \quad \sum_{v \in \mathcal{P}_n} \mathbb{1}_{\mathcal{L}}(v) = 1 \tag{1}$$
$$\forall v: \quad x(v) >= x(v^p)$$

where $\#(v)$ is the size (number of pixels) of segment (node) v, λ is a weighting value for the two energy terms, and $l(v_i)$ is the loss function defined for $v_i \in \{\mathcal{L}^-, \mathcal{L}^+\}$:

$$l(v_i) = \max(0, f(v_i) \cdot x(v_i)). \tag{2}$$

The loss function penalizes two contradictory cases: (i) segments in the group of under-segmented that receive positive scores; and (ii) segments in the group of over-segmented that receive negative scores. The problem will be solved via dynamic programming, as explained in the following section.

3.1.1 Inference by Dynamic Programming

The optimization problem in Equation 1 is highly structured and can be solved recursively by Dynamic Programming. For the subtree rooted at node v, its optimal slice $\mathcal{L}(v)$ is either the node v itself or the union of the optimal slices of all its child nodes v^c's, depending on whose energy is lower. Thus, the problem has optimal substructure [7] and so it naturally fits to the framework of dynamic programming to find the global optimal solution.

The problem proceeds from bottom to the top of the tree. For each subtree rooted at the current node v, the energy of $v \in \mathcal{L}(v)$ is computed and the energy of the optimal slices of all its child nodes is requested for comparison. The algorithm traverses back, and all comparison will be completed when the algorithm reaches the root node, and the global optimal of Equation 1 is obtained. The method is highly efficient with complexity $\mathcal{O}(N)$, where N is the total number of nodes. The global optimal of the energy can be found by applying Algorithm 1 to the root node, and the optimal slice is the corresponding set of nodes labeled to 0.

3.1.2 Predicting the Scales of Segments

In order to predict the scales (under-, properly-, or over-segmented) of the segments, we follow the route of modern computer vision systems to learn a predictor from human-annotated training data. To this end, we define a measure to compare the scale of an image segment $\mathbf{r}$ to that of the corresponding human-annotated segment $\mathbf{g}$. The correspondence is built up by computing the overlap between

Algorithm 1 Dynamic Programming in a Tree

[Yuhua:need to refine the algo]

Input: tree node v_i
if v_i is a leaf node **then**
  $\mathcal{C}_{v_i} \leftarrow \#(v_i) \cdot \max(0, f(v_i))$
  $E^*_{v_i} \leftarrow \#(v_i) \cdot \|f(v_i)\|^2$
else
  $\mathcal{C}_{v_i} \leftarrow \sum_{v_j \in \{v^c\}} \mathcal{C}_{v_j} + \#(v_i) \cdot \max(0, f(v_i))$
  $E^*_{v_i} \leftarrow min(\sum_{v_j \in \{v^c\}} E^*_{v_j} + \lambda \cdot \#(v_i) \cdot \max(0, -f(v_i)), \#(v_i) \cdot \|f(v_i)\|^2 + \lambda \cdot \sum_{v_j \in \{v^c\}} \mathcal{C}_{v_j})$
end if
return $\mathcal{C}_{v_i}, E^*_{v_i}$

computer-generated segments and human-annotated ones – the most-overlapping human-annotated segment is taken as the ground-truth of the computer-generated ones. The overlap is computed with the Intersection over Union (IoU).

After having the ground-truth segment $\mathbf{g}$, the scale of the segment $\mathbf{r}$ is then defined as:

$$S(\mathbf{r}) = \frac{\#(\mathbf{g}) - \#(\mathbf{r})}{\max(\#(\mathbf{r}), \#(\mathbf{g}))}. \tag{3}$$

The value of $S(\mathbf{r})$ is in $[-1, 1]$, with negative values for *under-*, 0 for *properly-* and positive values for *over-segmented* regions, the magnitude of the values representing the extent of being under- or over-segmented, which casts to what we expected from $f(v)$ (*c.f.* Section 3.1).

With Equation 3, the *scales* of the segments by segmentation methods can be computed and used as the training data to train our scale predictor.

As to the learning method, we employ a regression forest as the predictor $f(v)$. As to the features, we use a set of low-, and middle-level features, mainly following the work done for object proposals [3, 2]. The features are designed to capture a variety of region properties, and the detailed list of the features is provided in Sec 4.1.

The main difference between our prediction and the previous work [3, 25, 2] is that they predict the quality of segments, while we predict the scale of the segments. Although numerous measures have been proposed, it is still very hard to quantify the quality of segments. The granularity of segments, however, is easier to quantify, and it also provides more specific information such as under-segmented or over-segmented.

3.2. Hierarchy Re-scaling with Labeled Scales

After setting the optimal slice, we use it as an anchor slice to stretch the segmentation tree correspondingly. In our experiments, we use the threshold value of each optimal node as a control point, and linearly interpolate the original hierarchy.

Algorithm 2 Rescaling Hierarchy

Input: Optimal Slice $\mathcal{S}$, UCM map M_{ucm}
for $\mathbf{r} \in \mathcal{S}$ **do**
  $\mathbf{b} \leftarrow$ **Boundary**$(\mathbf{r})$
  $\mathbf{a} \leftarrow$ **InnerArea**$(\mathbf{r})$
  $m \leftarrow min(M_{ucm}(\mathbf{b}))$
  $M_{ucm}(\mathbf{a}) \leftarrow M_{ucm}(\mathbf{a}) * 0.5/m$
end for
$\mathbf{b_{all}} \leftarrow$ **Boundary**$(\mathcal{S})$
$m_{min} \leftarrow min(M_{ucm}(\mathbf{b_{all}}))$
$M_{ucm}(\mathbf{b_{all}}) \leftarrow m_{min} + \frac{(M_{ucm}(\mathbf{b_{all}}) - m_{min})}{2(1-m_{min})}$

Segmentation tree can be represented in the form of Ultra-Contour-Map(UCM) [1]. UCM is a matrix with the size $(2h-1)*(2w-1)$, where the h is the height of the original image, and w is the width. Therefore for each pair of neighboring pixels in the image, the value in the UCM matrix represents their boundary strength. And usually it is a value between 0 and 1. Segmentation of a certain scale can be extracted by thresholding the UCM with corresponding value. Without losing any generality, our algorithm directly manipulate on UCM due to its popularity and simplicity. The algorithm is summarized in Algorithm 2. We assume the functions **Boundary** to find the corresponding elements of boundary of a region $\mathbf{r}$ in the UCM matrix, and **InnerArea** to find its inner area.

By the presented algorithm, we perform locally linear transform on UCM map, and align the optimal slice to threshold 0.5, which makes it easier for later use. No information in the original hierarchy is lost during the rescaling process.

4. Experiments

We evaluate our approach on the segmentation hierarchies generated by multiple different segmentation methods, and further examine its usefulness on the task of object segmentation. The goal is to demonstrate that the proposed method is able to improve general segmentation hierarchies and the improvement is reflected to high-level vision tasks as well.

4.1. Experiment Settings

Dataset: We benchmark the performance of our approach on BSDS500 dataset [1], which includes 500 images, with 200 for training, 100 for validation, and 200 for testing. Each image is annotated by 5 different people on average. As to evaluation, we deploy three standard metrics: Segmentation Covering (SC), Probabilistic Rand Index (PRI), and Variation of Information (VI). Readers are referred to [1] for details about the dataset and the evaluation metrics.

Candidate methods: As to the candidate hierarchical segmentation methods, we chose the following four methods due to their popularity and good performance:

- gPb-owt-ucm [1]: a widely-used hierarchical segmentation method. Discriminative features are learned for local boundary detection and spectral clustering is applied on top of it for boundary globalization.
- MCG [2]: a unified framework for segmentation and object proposals. It combines information from multiple resolutions of the image and achieves the state-of-the-art results for both image segmentation and object proposals.
- SCG [2]: the single-resulution, faster version version of MCG. It gets competitive results at a fraction of the cost of MCG.
- PMI [15]: a recent work for unsupervised boundary detection. It can be applied for image segmentation as well in order to generate a hierarchical segmentation.

Training: The training set and the validation set of BSDS500 are pooled together as the training set for our regression forest. The four segmentation methods are used to generate hierarchies, over which the training samples (segments) are extracted. We train method-specific regression forests as the scale predictor. Since a large portion of regions in the hierarchies are very small and features extracted from them are not reliable, we exclude regions smaller than 50 pixels for the training of the predictor.

Specifically, for each region $\mathbf{r}$, we find its corresponding ground-truth region $\mathbf{g}$ by taking the human-annotated one with the highest covering score. The relative scale of $\mathbf{r}$ is then computed with Equation 3 for the regression target of $\mathbf{r}$. As to the features for $\mathbf{r}$, we draw on the success of object proposals [3, 2]. There, a large pool of middle-level features have been defined for segment description. The features used are summarized as follows:

- Graph partition properties: cut, ratio cut, normalized cut, unbalanced normalized cut.
- Region properties: area, perimeter, bounding box size, major and minor axis lengths of the equivalent ellipse, eccentricity, orientation, convex area, Euler number.
- Gestalt properties: inter- and intra-region texton similarity, inter- and intra-region brightness similarity, inter- and intra-region contour energy, curvilinear continuity, convexity.

Readers are referred to [3] for the details of these features. This list is definitely not exhaustive. More high-level features, *e.g.* by object detection, could be added.

Although these features are simple, extracting them for all layers of the segmentation hierarchies can be costly. Doing so is also unnecessary, as most segments at one layer are very similar to those from the parent layer and the child layer. Thus, we only extract the features from a subset of layers sub-sampled from the hierarchies. The layers are uniformly sampled over the range of UCM values.

As to the parameters of our method, we set 100 trees for the random forest. λ in Equation 1 is set to 0.1 to balance information from the three groups, because there are more segments over and under the optimal slice $\mathcal{L}$.

4.2. Results

The results of our method evaluated on top of the four candidates segmentation approaches are summarized in Table 1. As shown in the table, the improvements achieved by our alignment are considerable and, more importantly, they are consistent across different methods. The method improves more on ODS than OIS, this is because OIS accesses the ground-truth segmentations to search for the best-performing threshold, which somehow diminish the learned knowledge. We argue that ODS is more practical than OIS in a real vision systems, because for real applications there is no human-annotated segmentations.

Figure 6 shows qualitative results of different hierarchies. Our approach shows a consistent improvement over the original results. Since our approach if scale-aware, regions are of similar scale across images after alignment. Thus our method demonstrates better ability of preserving regions across images. In Figure 4 we show segmentation examples of MCG and aligned MCG by our method. As the figure shows, the aligned hierarchies generate characteristics closer to what human expect when flat segmentations are sampled out of the hierarchies. More particularly, after alignment, sampled segmentations of the hierarchies generate consistent responses across all parts of the image: all parts under-segmented, to all parts properly-segmented, and finally to all over-segmented while sampling from the top to the bottom of the hierarchies. This alignment greatly simplifies the use of hierarchical image segmentation for other vision tasks. Figure 6 shows qualitative results with different hierarchies. Our approach shows a consistent improvement over the original results. Since our approach is scale-aware, regions at the same level of the hierarchy are of similar scales across all areas of the images after the alignment. Thus our method demonstrates better ability of preserving regions across images.

We also tested the method in the scenario where the random forests are trained with segments from all of the four methods, and applied to all of them at test time. This gives slightly poorer results but in turn shows that our method can be applied in a method-agnostic approach.

	Covering (↑)		PRI (↑)		VI (↓)	
	ODS	OIS	ODS	OIS	ODS	OIS
MCG	0.61	0.67	0.83	0.86	1.57	1.39
MCG-aligned	0.63	0.68	0.83	0.86	1.53	1.38
SCG	0.60	0.66	0.83	0.86	1.63	1.42
SCG-aligned	0.61	0.67	0.83	0.86	1.61	1.41
gpb	0.59	0.65	0.83	0.86	1.69	1.48
gpb-aligned	0.60	0.66	0.83	0.86	1.66	1.46
PMI	0.53	0.59	0.76	0.81	2.03	1.80
PMI-aligned	0.54	0.59	0.76	0.81	2.01	1.80

Table 1. The results of our aligned hierarchies with a comparison to the original hierarchies.

	Covering (↑)		PRI (↑)		VI (↓)	
	ODS	OIS	ODS	OIS	ODS	OIS
Ncut [30]	0.45	0.53	0.78	0.80	2.23	1.89
Felz-Hutt [12]	0.52	0.57	0.80	0.82	2.21	1.87
Mean Shift [6]	0.54	0.58	0.79	0.81	1.85	1.64
Hoiem [14]	0.56	0.60	0.80	0.77	1.78	1.66
gPb-owt-ucm [1]	0.59	0.65	0.83	0.86	1.69	1.48
ISCRA [27]	0.59	0.66	0.82	0.85	1.60	1.42
PFE+mPb [34]	0.62	0.67	0.84	0.86	1.61	1.43
PFE+MCG [34]	0.62	0.68	0.84	0.87	1.56	1.36
MCG [2]	0.61	0.67	0.83	0.86	1.57	1.39
MCG+Ours	0.63	0.68	0.83	0.86	1.53	1.38

Table 2. Segmentation results on BSDS500 test set, with a comparison to the state-of-the-art competitors.

4.3. Comparison to Other Methods

As the previous section shows, the MCG aligned by our method generally performs the best. Here, we compare MCG-aligned to other competing methods. The results are summarized in Table 2 and demonstrate that segmentation quality can be improved by our alignment. In particular, the aligned MCG achieves the best result in Covering and VI. After alignment, the results are on par with the newest method of PFE+MCG [34]. It is noteworthy that our method and theirs are complementary, and the combination of the two may yield even better results. Their method is to improve feature embedding for a better local distance measure, while we aim to improve the hierarchy of existing segmentation methods.

4.4. Evaluation towards Object Segmentation

Segmentation *per se* is rarely the final objective of real applications, it is rather a middle tool towards, for instance, object segmentation [2] or semantic segmentation [17]. This section is devoted to show that better aligned hierarchies also help in this scenario.

In this case we firstly perform the evaluation using the object annotations provided on the BSDS300 set by [10] (we retrain on only BSDS300 train instead of BSDS500).

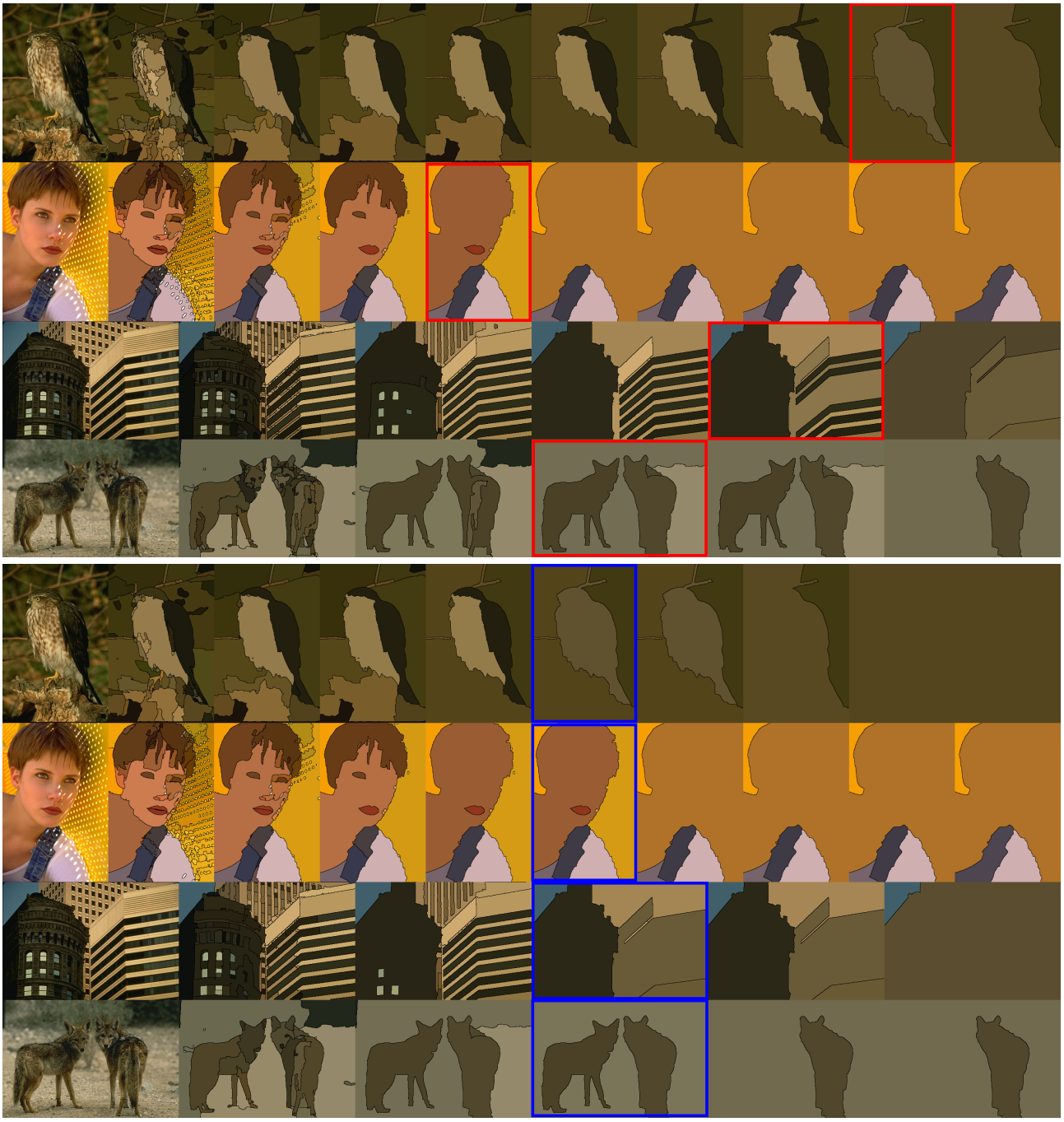

Figure 4. Results of MCG(first row) and MCG results improved by our approach(second row). Original images are shown in the left most. Segmentations of optimal-dataset-sclae(ODS) are given in the middle. And from left to right are different scales, from fine to coarse. Red bounding box indicates the scale with best results achieved by MCG, and blue box for ours. It can be seen that our approach provides better alignment, both across images and within one image.

The intuitive idea is to measure how well we can segment these objects by *selecting* regions from the different flattened hierarchies.

Figure 5 shows the achievable quality that an oracle could reached if selecting the regions from the original hierarchies or the ones with our newly-proposed alignment. The X axis corresponds to the number of needed regions, *i.e.*, the lower the better.

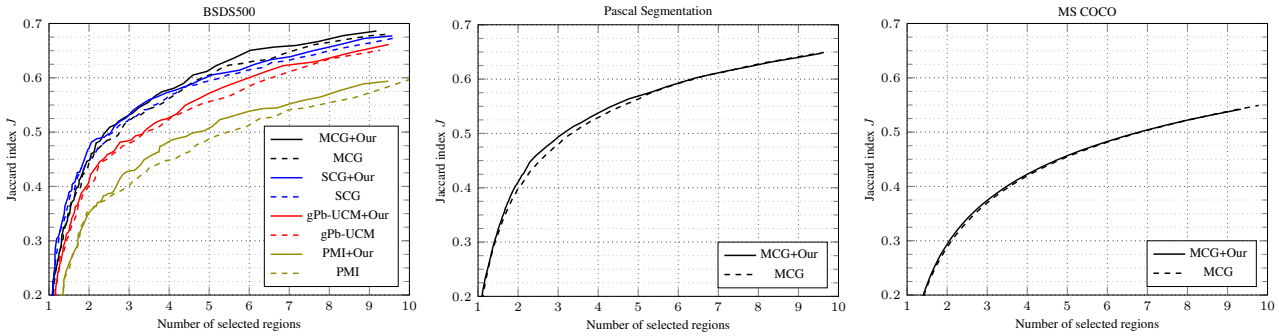

Figure 5. **Flattened hierarchies for object detection**: Achievable quality by an oracle with respect to the number of regions needed

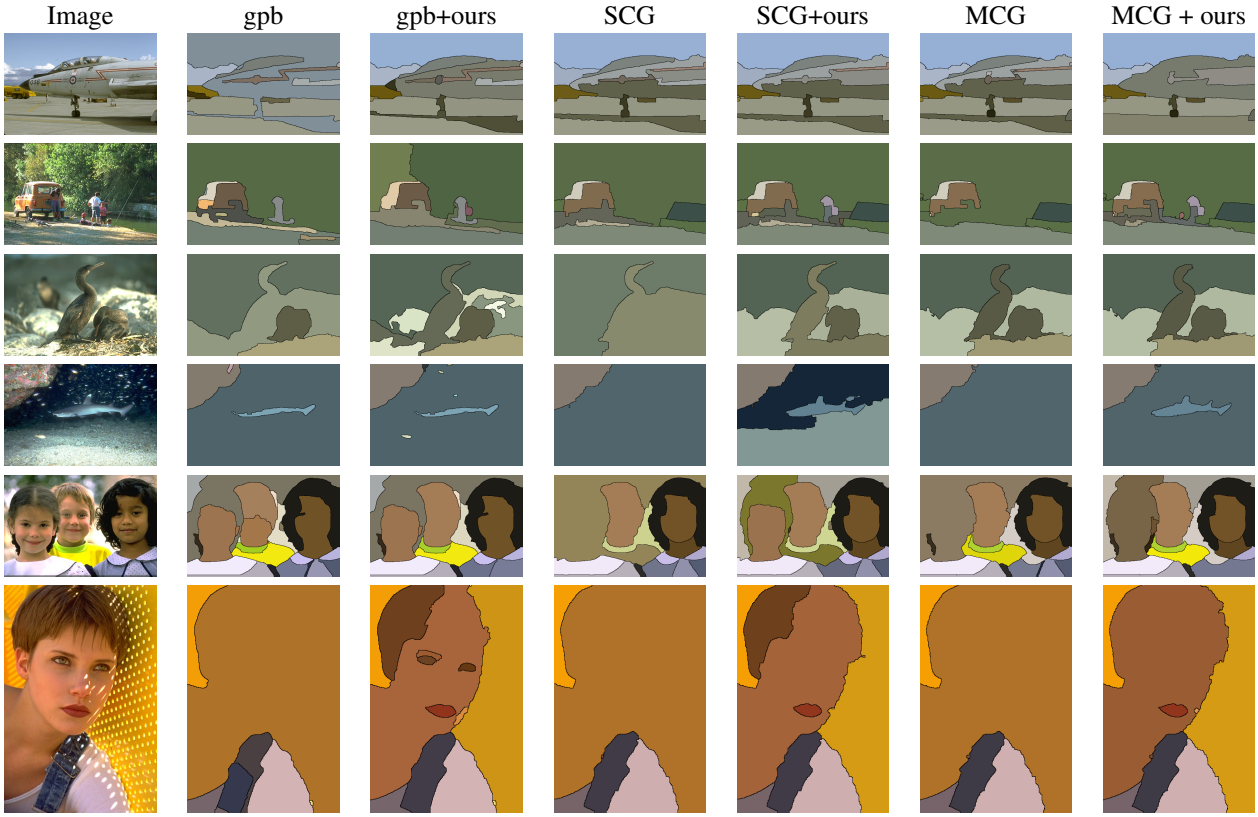

Figure 6. Comparison of segmentation results, hierarchies are flattened by optimal dataset scale(ODS).

We can observe that the aligned hierarchies consistently need less regions to get the same quality in all the tested hierarchies. In PMI, for instance, we need to select 5 regions to achieve the same quality that we can get with 4 on the aligned hierarchy. The combinatorial space of all possible 4-region combinations is significantly smaller and thus the search is more probable to succeed. On the other direction, if we limit the number of regions we get improvements up to 3 points (9%) in the achievable quality.

To further illustrates the scalability of the hierarchy alignment on larger dataset, we evaluated our alignment algorithm on Pascal VOC 2012 Segmentation set [11] and Microsoft COCO [19]. We retrain our scale predictor using the training set of Pascal 2012. In Pascal 2012 dataset, only the segmentation of foreground objects are given, in contrast to BSDS which is fully annotated. Thus during training we only consider all the segments that have overlap with foreground object annotations. The scale predictor is trained as described in Sec 3.1.2, the only difference is that $\mathbf{g}$ can only be foreground object. This strategy introduces extra bias towards foreground objects, because no information about the scale of background is given in the training process. However, we are still able to improve alignment of segmentation hierarchies . As shown in Fig**[Yuhua:add the figure]**,

we see that for the range of 2-3 regions (the one in which the MCG object proposal work), the aligned hierarchy provides a 2.5-point improvement ($\sim$6%), which shows that our method generalizes to larger datasets. **[Yuhua:describe some coco results]**

4.5. Running Time

Our approach takes approximately 3 seconds in total for each image, of which 2.39 seconds are spent on feature extraction from the segments. The prediction of regression forest takes about 0.45 seconds, and the dynamic programming takes 0.05 seconds for the inference. Finally, 0.11 seconds are spent for re-scaling the UCM. All times are measured on a standard desktop machine.

5. Conclusion

In this work, we presented a novel technique to align segmentation hierarchies. We proposed a method to learn and predict the scale of segments. We formulated the scale prediction for the segments in a hierarchy as a graph label problem, which is solved by dynamic programming. With the labeled scales as constraints, we then re-align the segmentation hierarchies by stretching the UCM maps. The method is evaluated on four different segmentation hierarchies on BSDS500, and it consistently improves their quality. We also showed that the improvement of segmentation hierarchies by our alignment is reflected well to a higher-level task of getting object segmentations.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE TPAMI*, 33(5):898–916, 2011. 1, 2, 5, 6

[2] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, pages 328–335. IEEE, 2014. 1, 2, 3, 4, 5, 6

[3] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, pages 3241–3248. IEEE, 2010. 3, 4, 5

[4] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. *arXiv preprint arXiv:1511.03339*, 2015. 3

[5] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603 –619, may 2002. 1

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002. 6

[7] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009. 4

[8] D. Dai, Y. Wang, Y. Chen, and L. Van Gool. Is image super-resolution helpful for other vision tasks? In *WACV*, 2016. 3

[9] I. Endres and D. Hoiem. Category-independent object proposals with diverse ranking. *IEEE TPAMI*, 36(2):222–234, 2014. 3

[10] I. Endres and D. Hoiem. Category-independent object proposals with diverse ranking. *IEEE TPAMI*, 36(2):222–234, 2014. 6

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 8

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 1, 6

[13] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, volume 1, pages 654–661. IEEE, 2005. 3

[14] D. Hoiem, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *IJCV*, 91(3):328–346, 2011. 6

[15] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, pages 799–814. Springer, 2014. 2, 5

[16] T. H. Kim, K. M. Lee, and S. U. Lee. Learning full pairwise affinities for spectral segmentation. *IEEE TPAMI*, 35(7):1690–1703, 2013. 1, 2

[17] V. Lempitsky, A. Vedaldi, and A. Zisserman. A pylon model for semantic segmentation. In *NIPS*, 2011. 1, 6

[18] J. Li, X. Liang, S. Shen, T. Xu, and S. Yan. Scale-aware fast r-cnn for pedestrian detection. *arXiv preprint arXiv:1510.08160*, 2015. 3

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 8

[20] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. 2007. 3

[21] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, volume 2, pages 416–423, 2001. 1

[22] S. E. Palmer. *Vision science: Photons to phenomenology*, volume 1. MIT press Cambridge, MA, 1999. 3

[23] B. Peng and O. Veksler. Parameter selection for graph cut based image segmentation. In *BMVC*, volume 32, pages 42–44, 2008. 3

[24] J. Pont-Tuset and F. Marques. Supervised assessment of segmentation hierarchies. In *ECCV*, pages 814–827. Springer, 2012. 3

[25] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, pages 10–17. IEEE, 2003. 3, 4

[26] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *CVPR*, 2013. 1, 2

[27] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *CVPR*, pages 2011–2018. IEEE, 2013. 6

[28] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, volume 2, pages 1605–1614. IEEE, 2006. 3

[29] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE TIP*, 9(4):561–576, 2000. 1

[30] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000. 1, 6

[31] J. Sun and H. Ling. Scale and object aware image thumbnailing. *IJCV*, 104(2):135–153, 2013. 3

[32] D. Varas, M. Alfaro, and F. Marques. Multiresolution hierarchy co-clustering for semantic segmentation in sequences with small variations. In *ICCV*, 12/2015 2015. 1

[33] C. Xu, S. Whitt, and J. J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, pages 2240–2247. IEEE, 2013. 1, 3

[34] Y. Yu, C. Fang, and Z. Liao. Piecewise flat embedding for image segmentation. 2015. 6