**Selective Search:**

We introduce Selective Search which combines the strength of both an exhaustive search and segmentation. Like segmentation, we use the image structure to guide our sampling process. Like exhaustive search, we aim to capture all possible object locations. Instead of a single technique to generate possible object locations, we diversify our search and use a variety of complementary image partitioning’s to deal with as many image conditions as possible

Inspired by bottom-up segmentation, we aim to exploit the structure of the image to generate object locations. Inspired by exhaustive search, we aim to capture all possible object locations. Therefore, instead of using a single sampling technique, we aim to diversify the sampling techniques to account for as many image conditions as possible. Specifically, we use a data-driven grouping based strategy where we increase diversity by using a variety of complementary grouping criteria and a variety of complementary colour spaces with different invariance properties. The set of locations is obtained by combining the locations of these complementary partitioning’s.

“””In this section we detail our selective search algorithm for object recognition and present a variety of diversification strategies to deal with as many image conditions as possible. A selective search algorithm is subject to the following design considerations:

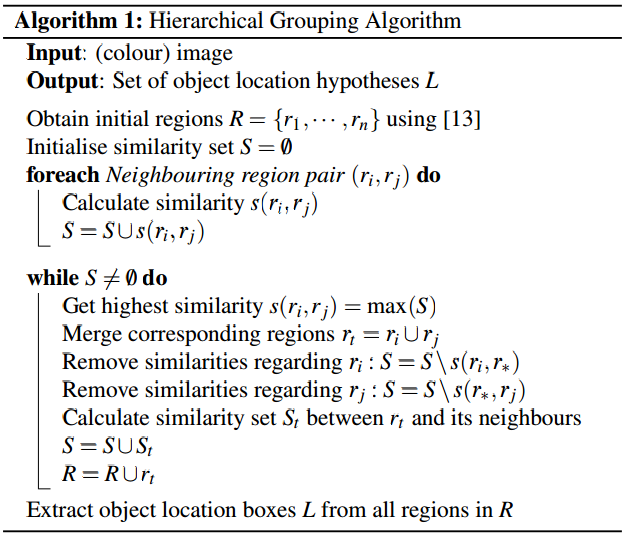
*Capture All Scales*. Objects can occur at any scale within the image. Furthermore, some objects have less clear boundaries then other objects. This is most naturally achieved by using an hierarchical algorithm.

*Diversification*. Therefore instead of a single strategy which works well in most cases, we want to have a diverse set of strategies to deal with all cases.

*Fast to Compute.”””*

*Selective Search by Hierarchical Grouping:*

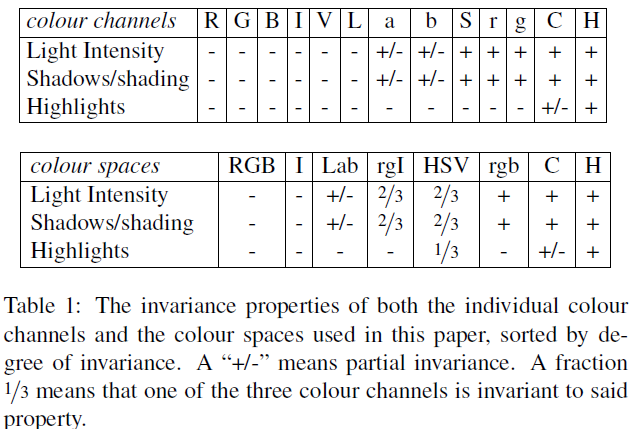
Our grouping procedure now works as follows. We first use [13] to create initial regions.



For the similarity s(ri ,rj) ,In effect, this means that the similarities should be based on features that can be propagated through the hierarchy, i.e. when merging region ri and rj into rt , the features of region rt need to be calculated from the features of ri and rj without accessing the image pixels

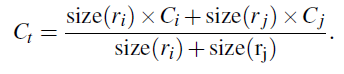
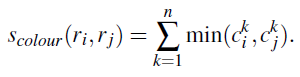
Diversification: Need to read all the text provided as it is quite important.

1) Complementary Colour Spaces:

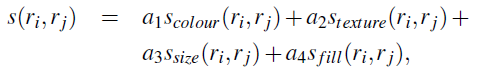
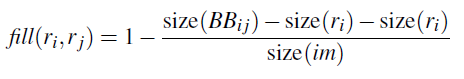
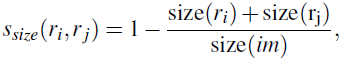
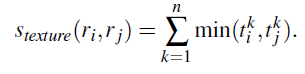


2) Complementary Similarity Measures:

*scolour*(*ri*, *r j*) measures colour similarity. Specifically, for each region we obtain one-dimensional colour histograms for each colour channel using 25 bins, which we found to work well. This leads to a colour histogram *Ci* = {*c*1 *i* , · · · ,*cni* } for each region *ri* with dimensionality *n* = 75 when three colour channels are used. The colour histograms are normalised using the *L*1 norm. Similarity is measured using the histogram intersection:



Similarly, we calculate other similarities:



*sfill*(*ri*, *r j*) measures how well region *ri* and *r j* fit into each other. The idea is to fill gaps: if *ri* is contained in *r j* it is logical to merge these first in order to avoid any holes. On the other hand, if *ri* and *r j* are hardly touching each other they will likely form a strange region and should not be merged.

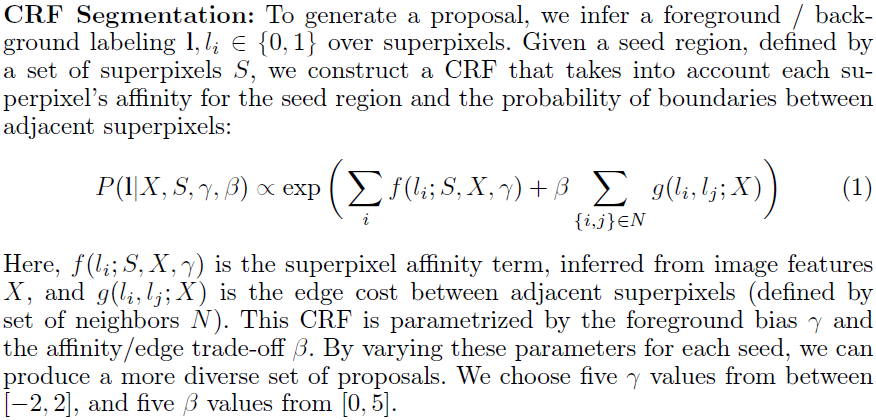
3) Complementary Starting Regions:

We choose to order the combined object hypotheses set based on the order in which the hypotheses were generated in each individual grouping strategy. However, as we combine results from up to 80 different strategies, such order would too heavily emphasize large regions. To prevent this, we include some randomness as follows. Given a grouping strategy *j*, let *r j* *i* be the region which is created at position *i* in the hierarchy, where *i* = 1 represents the top of the hierarchy (whose corresponding region covers the complete image). We now calculate the position value *vj* *i* as RND×*i*, where RND is a random number in range [0,1]. The final ranking is obtained by ordering the regions using *vj* *i* .

**Category Independent Object Proposals**

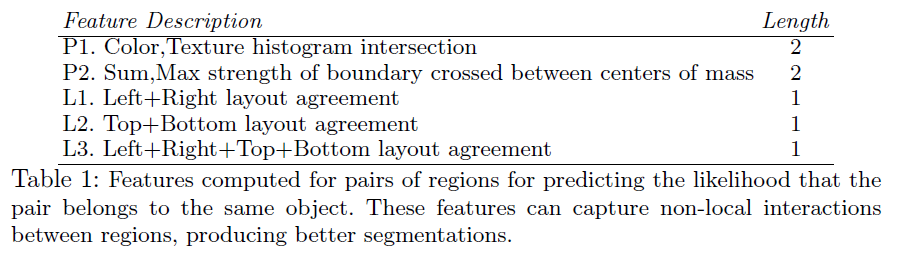
\*We first generate a large and diverse bag of proposals that are directed to be more likely to be object regions. Each proposal is generated from a binary segmentation, which is seeded with a sub region of the image. This seed is assumed to be foreground, and a segmenter selects pixels likely to belong to the same foreground object as the seed.

Category independent object proposals attempts to provide region proposals through the use of hierarchical segmentation followed by a ranking of the calculated segments within the image. Hierarchical segmentation is the starting mechanism for this algorithms, and is carried out using Hoiem et al. [4]. Seeds are points which serve as a starting location for object proposals. The seeds are generally identified on the basis of colour and texture features.



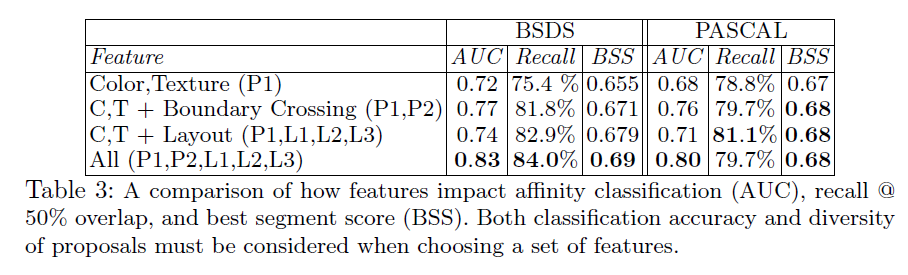
In the above expression, it is necessary to calculate the superpixel affinity term. We first compute each region R's affinity for lying on the same object as the seed S. We learn the foreground probability P(lRjS;X) with a boosted decision tree classifier.

Boundary cues are encoded by considering the cost to pass across boundaries from one region to the other. This path across boundaries is the straight line between their centres of mass.

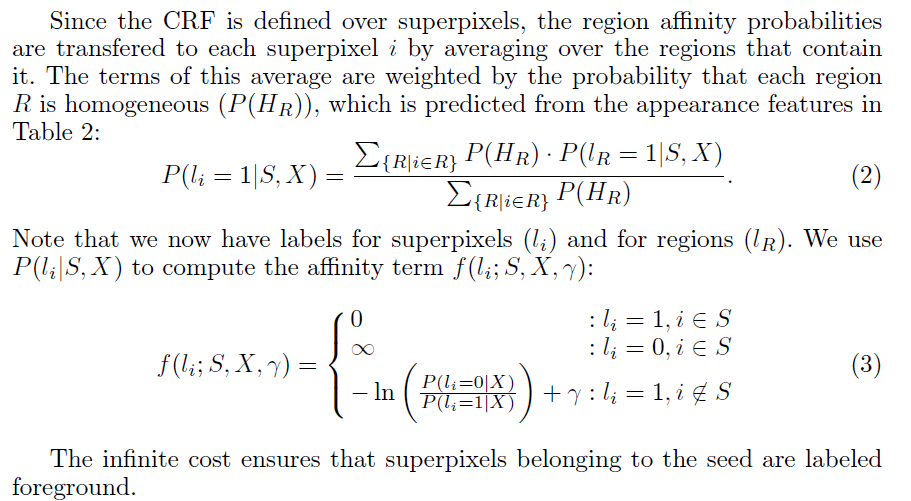


By setting a cost of crossing over boundaries, we can make sure that the size of the segments is not too small and limited in its features.

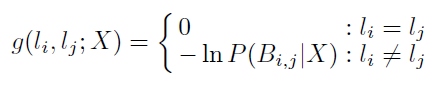
Another layout feature introduced to improve object proposal is determining the location of the object in the region of proposal. These predictions are made by boosted decision tree classifiers based on histograms of occlusion boundaries, where the boundaries are separated based on figure/ground labels. As a feature, we measure whether the layout predictions for two regions are consistent with them being on the same object.



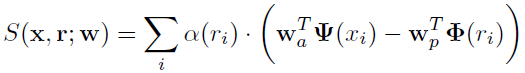
Note: best segmentation overlap score for each object (BSS)



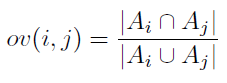
Finally, there is an edge cost which allows us make strong and distinct boundaries, by penalizing superpixels given different values when their boundary is weak. Given below is the formula-

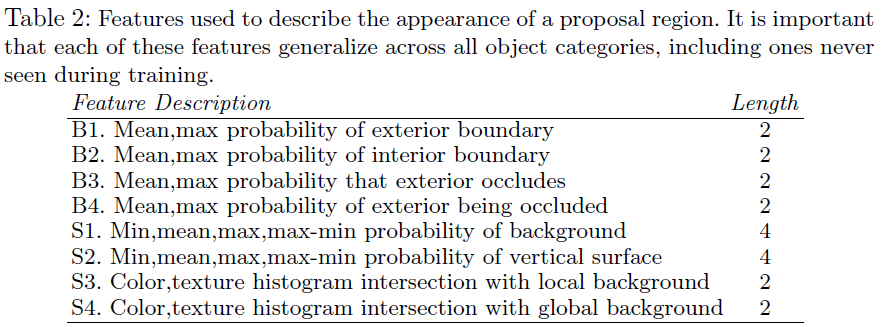
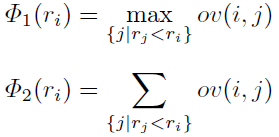


We now have a bag of proposals for object proposals. We now rank these proposals, assigning the most diverse of these the highest ranks, in order to detect ‘all’ the objects in the image.



The score is a combination of appearance features (x) and overlap penalty terms (r), where r indicates the rank of a proposal, ranging from 1 to the number of proposals M. This allows us to jointly learn the appearance model and the trade-off for overlapping regions. \_1(r) penalizes regions with high overlap with previously ranked proposals, and 2(r) further suppresses proposals that overlap with multiple higher ranked regions. The second penalty is necessary to continue to enforce diversity after many proposals have at least one overlapping proposal:





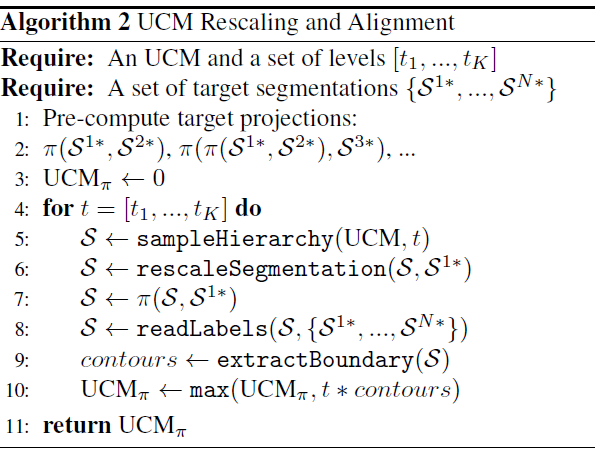
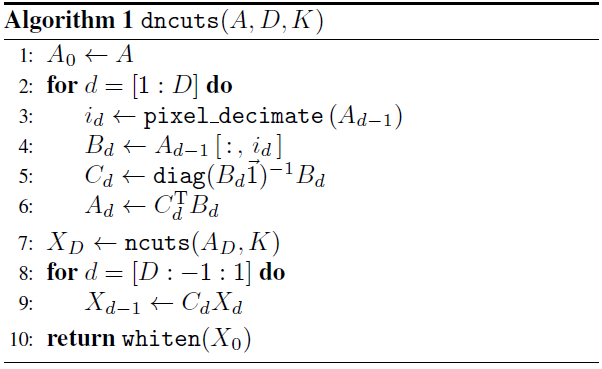
In order to find the highest ranked proposals, rather than the most violated proposals (overlapping), we use a ‘learning’ based model. This allows us to improve performance over iterations.

**Multiscale Combinatorial Grouping**

Consider a segmentation of the image into regions that partition its domain S = {Si}i. A segmentation hierarchy is a family of partitions {S⇤, S1, ..., SL} such that: (1) S⇤ is the finest set of superpixels, (2) SL is the complete domain, and (3) regions from coarse levels are unions of regions from fine levels.

A hierarchy where each level Si is assigned a real-valued index !i can be represented by 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 !i in the UCM produces the segmentation Si.



Multiscale Hierarchal 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.

Hierarchy Alignment We construct a multiresolution pyramid with N scales by subsampling / super sampling 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

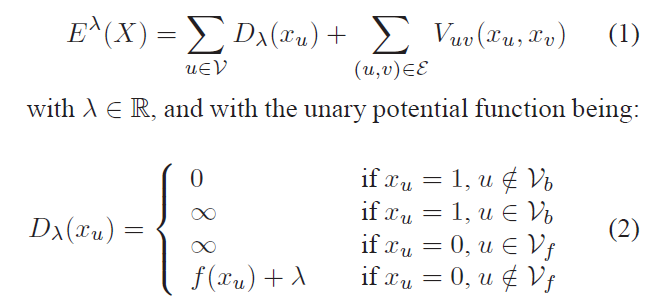
SN⇤ in the highest-resolution. We extract the finest superpixels of each hierarchy, rescale them to the original image resolution, pre-compute their successive projections to SN⇤ and then transfer recursively the strength of all the coarser UCMs by applying Algorithm(2). This is quite similar to what is carried out in HOG, and can be considered analogous in its approach.

Once we have all the boundaries, it is necessary find the probability of them being in a similar boundary. 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.

At its core are a fast eigenvector computation for normalized-cut segmentation and an efficient algorithm for combinatorial merging of hierarchical regions.

**Constrained Parametric Min-Cuts:**

Let I(V) ! R3 be an image defined on a set of pixels V. As commonly done in graph-based segmentation algorithms, the similarity between neighbouring pixels is measured and encoded as edges of a weighted graph G = (V, E). Here, we use each pixel as a node and augment the node set V with two special nodes s and t that are required to be in separate partitions for any binary cut. They identify the foreground and the background, respectively. Given foreground and background seed pixels Vf and Vb, our overall objective is to minimize over pixel labels {x1, ..., xk}, xi 2 {0, 1}, with k being the total number of pixels, the following energy function:



*Grid Geometry:*Hence for all our experiments, we report results obtained with foreground seeds placed on a regular 5 × 5 grid. The background seeds are required in order to prevent trivial cuts that leave the background set empty. We used four different types: covering the full image boundary, just the vertical edges, just the horizontal edges and all but the bottom image edge. This is to allow for objects that are only partially inside the image.

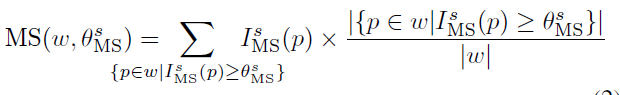
*Fast Rejection:* We first filter out very small segments (up to 150 pixels in our implementation), then sort the segments using a simple energy (we use the ratio cut [33] as the energy because it is scale invariant and very effective) and keep the top up to 2000 segments. Then we hierarchically cluster the segments using their overlap as similarity, in order to form groups where all segments have at least 0.95 overlap. For each cluster, we keep the segment having the lowest energy.

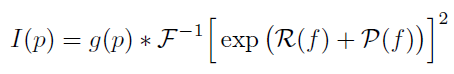
*Ranking Segments*: 1) **Graph partition properties (8 features)** include the cut (sum of affinity along the segment boundary) [35], the ratio cut (sum of affinity along the boundary divided by their number) [33], the normalized cut (ratio of cut and affinity inside foreground, plus ratio of cut and affinity in background) [31], the unbalanced normalized cut (cut divided by affinity inside foreground) [30], and if the fraction of the cut larger than a threshold, normalized by segment perimeter, for 4 different thresholds. B) **Region properties (18 features)** include area, perimeter, relative coordinates of the region centroid in the image, bounding box location and dimensions, major and minor axis lengths of the ellipse having the same normalized second central moments as the region, eccentricity, orientation, convex area, Euler number, diameter of a circle with the same area as the region, ratio of pixels in the region to pixels in the total bounding box, perimeter and absolute distance to the centre of the image.C) **Gestalt properties (8 features)** include inter-region texton similarity (chi-square distance between bag of textons computed over foreground and background), intra-region texton similarity (number of different textons present in foreground in quantity larger than a particular fraction of the area of the segment), inter-region brightness similarity (chisquare distance between intensity histograms of foreground and background regions), intra-region brightness similarity (measured in a similar way to texton homogeneity), interregion contour energy (sum of edge energy inside foreground region, computed using globalPb, normalized by perimeter), intra-region contour energy (sum of edge energy along boundary, normalized by perimeter), curvilinear continuity (sum of consecutive angle differences of the line segment approximating the contour), convexity (ratio of area of foreground region and its convex hull).

**Objectness:**

**Objectness Cues:** These are features analysed to identify objects in the frame. The following four features have been discussed:

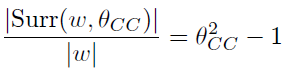
1. Multi Scale Saliency:





Saliency (Formula 2) is higher for windows with higher density of salient pixels (second factor), with a bias towards larger windows (first factor). Density alone would score highest windows comprising just a few very salient pixels. Instead, our measure is designed to score highest windows around entire blobs of salient pixels, which correspond better to whole objects.

1. Colour Contrast:



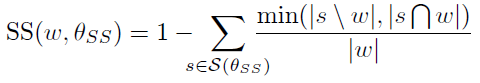
1. Edge Density:

The binary edgemap IED(p) 2 f0; 1g is obtained using the Canny detector [8], and Len(\_) measures the perimeter of the inner ring. The ED cue captures the closed boundary characteristic of objects, as they tend to have many edgels in the inner ring.

1. Super Pixels Straddling(SS):

Superpixels segment an image into small regions of uniform color or texture. A key property of superpixels is to preserve object boundaries: all pixels in a superpixel belong to the same object [46] (ideally). Hence, an object is typically oversegmented into several superpixels, but none straddles its boundaries (fig. 5). Based on this property, we propose here a cue to estimate whether a window covers an object.

The SS cue measures for all superpixels s the degree by which they straddle w:



Superpixels entirely inside or outside w contribute 0 to the sum. For a straddling superpixel s, the contribution is lower when it is contained either mostly inside w, as part of the object, or mostly outside w, as part of the background (fig. 5c). Therefore, SS(w; \_SS) is highest for windows w fitting tightly around an object, as desired.

We present here an efficient procedure to reduce this complexity. For each superpixel s we build a separate integral image Is(x; y) giving the number of pixels of s in the rectangle (0; 0) ! (x; y). Then, using Is(x; y), we can compute the number js T wj of pixels of s contained in any window w in constant time, independent of its area (fig. 7). The area js nwj outside w is also readily obtained as:



1. Size and Location:

We compute the probability of an image window to cover an object using kernel density estimation [44] in the 4D space W of all possible windows in an image. The space W is parametrized by the (x; y) coordinates of the center, the width, and the height of a window. We equip W with a probability measure pW defining how likely a window is to cover an object. This probability pW is computed by kernel density estimation on a large training set of N windows fw1;w2; : : : ;wNg covering objects, which are points in W. As training images might have different sizes, we normalize the coordinate frame of all images to 100 \_ 100 pixels.

**Learning Cue Parameters:**