Prime Object Proposals with Randomized Prim's Algorithm

Santiago Manen[1] Matthieu Guillaumin[1] Luc Van Gool[1,2]

[1]Computer Vision Laboratory
ETH Zurich

[2]ESAT - PSI / IBBT
K.U. Leuven

{smanenfr, guillaumin, vangool}@vision.ee.ethz.ch

Abstract

Generic object detection is the challenging task of proposing windows that localize all the objects in an image, regardless of their classes. Such detectors have recently been shown to benefit many applications such as speeding-up class-specific object detection, weakly supervised learning of object detectors and object discovery.

In this paper, we introduce a novel and very efficient method for generic object detection based on a randomized version of Prim's algorithm. Using the connectivity graph of an image's superpixels, with weights modelling the probability that neighbouring superpixels belong to the same object, the algorithm generates random partial spanning trees with large expected sum of edge weights. Object localizations are proposed as bounding-boxes of those partial trees.

Our method has several benefits compared to the state-of-the-art. Thanks to the efficiency of Prim's algorithm, it samples proposals very quickly: 1000 proposals are obtained in about 0.7s. With proposals bound to superpixel boundaries yet diversified by randomization, it yields very high detection rates and windows that tightly fit objects.

In extensive experiments on the challenging PASCAL VOC 2007 and 2012 and SUN2012 benchmark datasets, we show that our method improves over state-of-the-art competitors for a wide range of evaluation scenarios.

1. Introduction

Generic object detection is a recent development of computer vision research that has received a fast-growing interest [1, 3, 9, 27], cf Fig. 1. This is mainly due to the large number of applications of such systems. The most common motivation for using such object proposals relates to class-specific object detection [2, 27]. The proposals can indeed be used as a replacement for the computationally expensive sliding window approach for object detection [6, 12, 15]. This yields important computational savings as the number of classes and the complexity of the object detectors grow.

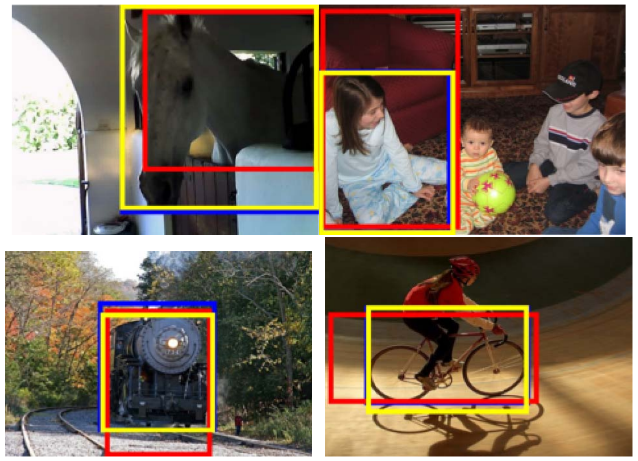


Figure 1: Detecting *any* object in images (bounding-boxes in blue) is a recent development of computer vision. We compare the best proposals of our approach (in yellow) with Objectness ([2], in red) on four images of the VOC2007 dataset. Our algorithm generates tighter-fitting windows. [NB: Figures best viewed in color.]

Clearly, the computational cost to obtain the proposals is of crucial importance as it should not imper the later savings.

Generic object proposals are also used as a regularization for weakly supervised learning approaches in vision. By limiting the set of possible object locations to those that are likely to contain an object, it becomes possible to learn the appearance and localize objects of new classes [7, 8, 16, 17, 24, 23]. In a similar spirit, recent work has explored applications in object discovery [18], weakly supervised learning of object interaction with humans [19] or with other objects [5], as well as action recognition in still images [22] and content-aware media re-targeting [26].

For all of these applications, the quality of the underlying object proposals – commonly measured by the intersection-over-union (*IoU*) [10] of image windows – is a critical factor of performance [21]. Indeed, when considering object detection, the recall of the complex models applied to object proposals cannot exceed the recall of the proposals themselves. Moreover, for the application of weakly supervised learning, the quality of possible object locations will impact both the ability to find those objects and the quality of the models that can be learnt from those windows used as automatic annotations. This highlights the need for methods that are able to generate high-quality and tightly fitting windows, and, in particular, this implies performing analysis beyond the coarse detection criterion of 0.5 *IoU*.

In this paper, we propose a novel algorithm to generate very quickly high-quality object proposals, *c.f.* Fig. 1. Our approach is based on Prim's algorithm [20], which greedily computes the maximum spanning tree of a weighted graph. The stochastic version we propose, the *Randomized Prim's* (RP) algorithm, is designed to sample random partial spanning trees of a graph with large expected sum of edge weights. This is done by (i) replacing the greedy selection of edges in Prim's algorithm with multinomial sampling proportional to edge weights, and (ii) using a randomized termination criterion to avoid covering the full graph.

To obtain the proposals, we apply RP on the graph induced by the superpixels [13] of an image, with edge weights representing the likelihood that two neighbouring superpixels belong to the same object. Based on a training set, we use logistic regression to discriminatively learn these weights as a linear combination of several superpixel similarities. When the randomized stopping criterion of RP is met, we generate an object proposal using the bounding-box of the superpixels spanned by the current tree.

Our approach combines the following advantages: (i) superpixel boundaries yield proposals that tightly fit objects; (ii) randomization increases the diversity of our proposals and (iii) RP is very efficient, leading to a very fast object proposal method. We have conducted extensive experiments on the PASCAL VOC2007 [10], VOC2012 [11] and SUN2012 [29] benchmark data sets and show the superiority of RP in speed and performance compared to state-of-the-art methods [2, 21, 27], especially in difficult scenarios such as *IoU* ≥ 0.6: 1000 object proposals are obtained in less than 0.7s while detecting 74% of the objects.

Below, we first discuss related work (Sec. 2), then describe RP in details (Sec. 3). We present how we use RP for object proposals in Sec. 4, including how we learn the edge weights for the graph of superpixels. We then present our experiments in Sec. 5 and draw conclusions in Sec. 6.

2. Related Work

Generic object detection is a fairly recent topic of research, with origins dating only three years ago [1, 3]. Alexe *et al.* [1] introduced the *Objectness* measure, which samples image windows with the probabilities that they contain an object of any class. This is performed using a set of well-designed cues that are combined in a Naïve Bayes framework. To sample from the 4d distribution of windows with scores, the authors propose to first score 100.000 windows, then subsample them based on their probabilities. In their subsequent work [2], significant improvements in detection rate were obtained using Non-Maximum Suppression as in class-specific object detection [12]. This approach proved very successful and triggered several extensions, including additional cues and using discriminative training [21], fusion with region saliency [4] and generalization to video using motion segmentation [25].

Carreira *et al.* [3] simultaneously proposed a method based on graphcuts to generate image segmentations that are likely to contain objects. Each segmentation generates a bounding-box that is used as proposal for an object. To obtain many proposals, several graphcuts are run using random positive and negative seeds. In a similar spirit, in [9], initial superpixels are grown using foreground-background CRF segmentation with random seeds, and proposals are ranked according to extracted features.

These existing methods have benefits over our approach. [2, 21] provide scores with the proposals, which improve detection rate (by re-ranking proposals) and serve as prior knowledge for discovering new object categories [8]. [3, 9] are based on image segmentation and hence provide a pixel-level mask for each proposal. However, they are typically slow (several seconds [2] to several minutes [3, 9] per image) and are not able to retrieve all the objects in all images.

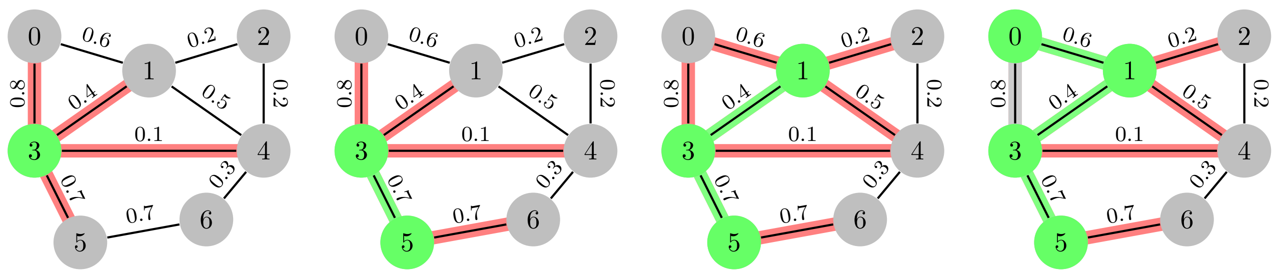
The current state-of-the-art method in terms of detection rate operates on the connectivity graph of an image's superpixels [27]. It performs an ad-hoc hierarchical bottom-up agglomeration of groups of superpixels. Groups are greedily merged two-by-two according to their similarities, and proposals are generated at each step of the agglomeration. The procedure yields a fixed number of proposals that is only twice the original number of superpixels in the image.

Contrary to the hard-decisions taken by [27], our approach is based on *randomly growing* groups of superpixels. This allows to generate any desirable number of object proposals and explore new groupings across several runs. Therefore, it shows more diversity in the set of object proposals and obtains significantly higher dectection rates than [27]. Moreover, because it does not compute similarities between groups of superpixels, our approach is significantly faster (about 6 times faster for 1000 proposals).

3. The Randomized Prim's Algorithm

We cast the problem of sampling connected groups of superpixels that are likely to contain the same object as that of sampling partial spanning trees of superpixels that have high sum of edge weights. This is the role of the *Randomized Prim's* (RP) algorithm that we present here.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \rho)$ be the weighted connectivity graph of the superpixel segmentation of an image, where the vertices $\mathcal{V}$ are the superpixels and the edges $(n, m) \in \mathcal{E}$ connect superpixels n and m. The weight function $\rho : \mathcal{E} \to [0, 1]$



(a) Initialization (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 2: (a) The Randomized Prim's algorithm initializes the tree T_1 (green) with a random node. At iteration k (b-d), a new edge is added to the tree T_k. The edges are sampled from the set $\mathcal{E}_k$ (red) of edges connecting T_k to its frontier, proportionally to their edge weights. The Prim's algorithm corresponds to always selecting the edge in $\mathcal{E}_k$ with maximum weight.

assigns weights $\rho(n,m) = \rho_{n,m}$ to edges. For now, we will assume that those weights represent the probability that the superpixels n and m belong to the same object.

For a vertex $n \in \mathcal{V}$, let $N(n) \subset \mathcal{V}$ be the subset of vertices connected to n, *i.e.* its neighbours. We denote as the *frontier* $N(S)$ of a set of vertices $S \subset \mathcal{V}$ the union of its neighbourhoods: $N(S) = \bigcup_{n \in S} N(n) \backslash S$.

Our algorithm generates random partial spanning trees independently. Thus, we describe below how a single tree is sampled using the Randomized Prim's algorithm, and this procedure is repeated as many times as required by the user.

3.1. Description of the Core Algorithm

Similar to the Prim's algorithm, the Randomized Prim's algorithm is an iterative tree-growing procedure. At each iteration k of the algorithm, we refer to the current partial spanning tree as T_k. We initialize the tree with a random vertex from the graph: $n \sim \mathrm{Unif}(\mathcal{V})$ and $T_1 = \{n\}$.

At each subsequent iteration, we sample a candidate superpixel from the frontier $N(T_k)$ to add to the tree T_k. To do so, we look at the edges $\mathcal{E}_k$ connecting T_k to its frontier: $\mathcal{E}_k = \mathcal{E} \cap (T_k \times N(T_k))$. An edge $(n,m) \in \mathcal{E}_k$ is then sampled from the multinomial distribution associated with probabilities proportional to $\rho_{n,m}$, which we denote as $\mathrm{Mult}(\mathcal{E}_k, \rho)$:

$$e_k = (n,m) \sim \mathrm{Mult}(\mathcal{E}_k, \rho). \tag{1}$$

This is equivalent to sampling the vertex m from $N(T_k)$ proportionally to the sum of weights of edges leading to m in $\mathcal{E}_k$, or, alternatively, to first sample uniformly a superpixel n in T_k and then sample m based on $\rho_{n,m}$. Fig. 2 illustrates this procedure. Note, Prim's algorithm instead greedily chooses the edge in $\mathcal{E}_k$ with the largest weight, *i.e.* $e_k = \mathrm{argmax}_{(n,m) \in \mathcal{E}_k} \rho_{n,m}$.

3.2. Multinomial Sampling over a Dynamic Set

The key technical element for the efficiency of Prim's algorithm is the data structure to keep track of the dynamic set of edges $\mathcal{E}_k$. As our graphs are planar and thus have low density,[1] a max-heap leads to a time complexity of $O(|\mathcal{V}| \log |\mathcal{V}|)$. With a max-heap, extracting the maximum element, inserting or deleting elements are $O(\log |\mathcal{E}_k|)$, *i.e.* logarithmic with respect to the size of the set of edges.

We adapt this data structure to obtain a similar time complexity for the Randomized Prim's algorithm. The key idea is that multinomial sampling can be performed using binary search on the cumulative sum of probabilities [28], and those probabilities need not be normalized. We therefore extend a binary search tree structure (BST) so as to maintain, for each node i, the sum of edge weights W_i of the subtree. Using this BST, we uniformly sample a number w between 0 and the total sum of edge weights W (readily accessible at the root node). Based on the values W_L and W_R of the children nodes, we perform a recursive binary search to find the edge e_k such that the sum w_k of edge weight on its left satisfies $w_k \leq w < w_k + \rho(e_k)$. When adding or removing nodes in the BST, one simply need to update W_i for all the ancestor nodes i. Hence the complexity of extraction of sampled edges, insertion and deletion are all $O(\log |\mathcal{E}_k|)$.

3.3. Randomized Stopping Criterion

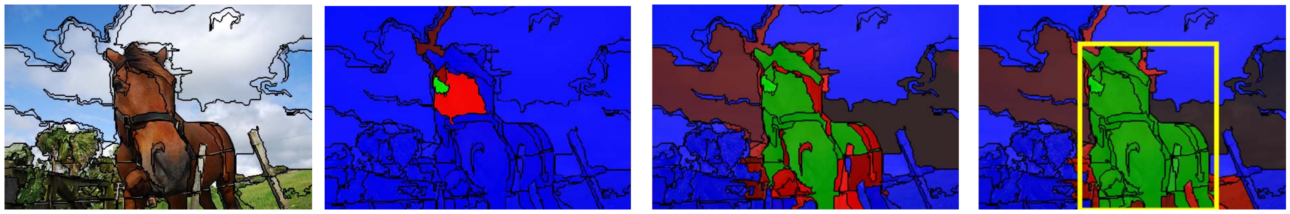
In RP, a stopping criterion is necessary in order to sample partial spanning trees that do not cover the full graph of superpixels. This allows to propose windows other than the full image. To do this, we sample a uniform stopping criterion ξ_0 between 0 and 1 at the initialization step and use a function $\xi(\cdot) \in [0,1]$ to evaluate the opportunity to add e_k to T_k, and terminate as soon as $\xi(T_k, e_k) > \xi_0$. Otherwise we add e_k to T_k and proceed with iterations. We detail in Sec. 4.2 our choice of $\xi(T_k, e_k)$ for object proposals.

We summarize in Alg. 1 the full procedure to sample a partial spanning tree with large expected sum of edge weights using the Randomized Prim's algorithm.

4. Randomized Prim's for Object Proposals

Using the connectivity graph of a superpixel segmentation [13], we obtain groups of connected superpixels by applying the Randomized Prim's algorithm, and we use the bounding-boxes of these groups as window proposals. We illustrate the process on an example image in Fig. 3.

[1]A planar graph with more than 3 vertices has at most $3|\mathcal{V}| - 6$ edges.



(a) Superpixel segmentation (b) Initialization (c) Iteration k (d) Proposal from partial tree

Figure 3: We apply the Randomized Prim's algorithm to the connectivity graph of superpixels of an image (a). (b) It starts with one superpixel (green). At each iteration (c), it samples a neighbouring superpixel (red) and decides to add it or return the bounding-box as a proposal (d). The brightness of red indicate the relative probability of sampling superpixels (lighter means more probable). The superpixels in blue are not connected to the current tree, hence cannot be sampled.

Algorithm 1: The Randomized Prim's Algorithm.

```
input : Weight graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \rho)$
output: Partial spanning tree $T_k$
$k = 0$;
$T_k = \varnothing$;
$\mathcal{E}_0 = \mathrm{BST}(\varnothing)$ ;          /* Empty binary search tree */
$m \sim \mathrm{Unif}(\mathcal{V})$;
$\xi_0 \sim \mathrm{Unif}(0, 1)$;
repeat
    $k = k + 1$;
    $T_k = T_{k-1} \cup \{m\}$;
    /* Add edges $(m, p)_{p \in N(T_k)}$, remove $(p, m)_{p \in T_k}$ */
    $\mathcal{E}_k = \mathrm{updateBST}(\mathcal{E}_k, m, T_k, N(T_k), \mathcal{E}, \rho)$;
    $(n, m) \sim \mathrm{Mult}(\mathcal{E}_k, \rho)$;
    $\xi(T_k, e_k) = (1 - \rho_{n,m} + \alpha(T_k))/2$;
until $\xi(T_k, e_k) > \xi_0$ ;
```

Following [27], we further augment the diversity of windows using $C = 4$ different graphs corresponding to the superpixel segmentations of 4 color spaces (HSV, Lab, Opponent and rg). To sample a single window proposal, we uniformly select one graph $\mathcal{G}_c$ among the C and proceed as in Alg. 1 using $\mathcal{G}_c$. The procedure is simply repeated as many times as the user desires.

4.1. Learning Edge Weights

An important aspect for RP is to set the weights $\rho_{n,m}$ from which edges (n, m) will be sampled. We model the weight ρ with a logistic function to obtain the probability that n and m contain the same object:

$$\rho_{n,m} = \sigma(\mathbf{w}^T \Phi_{nm} + b) \quad (2)$$

$$\sigma(x) = (1 + \exp(-x))^{-1}, . \quad (3)$$

where Φ_{nm} is a vector containing simple and efficient features that measure the similarity and compatibility of n and m, σ is the sigmoid function and b a bias term.

We resort to training data to learn the weights and bias. For this, we assign a superpixel to a segmented object if at least 60% of its surface is within the object. Then we mine for pairs of superpixels that belong to the same object (positive pairs, $y_i = 1$), and pairs that do not (negative pairs, $y_i = 0$) and compute their feature vectors Φ_i. We use the maximum likelihood estimator to set $\mathbf{w}$ and b:

$$\{\mathbf{w}^*, b^*\} = \underset{w,b}{\operatorname{argmax}} \sum_i y_i \ln \rho_i + (1 - y_i) \ln(1 - \rho_i), \quad (4)$$

where $\rho_i = \sigma(\mathbf{w}^T \Phi_i + b)$. This log-likelihood function is concave, so we simply perform gradient ascent.

The features we have combined in Φ are simple features than can be computed efficiently:

1. Color Similarity f_c. Color consistency is a important cue for objects. With h_n the normalized color histogram of the superpixel n, $f_c(n, m) \in [0, 1]$ is set as the l_1-norm of their intersection: $f_c(n, m) = |h_n \cap h_m|$. We have used the Lab colorspace and 16 bins for each component.

2. Common Border Ratio f_b. Connections between superpixels are not all as likely to happen within objects. Let l_n and l_m be the perimeters of superpixels n and m (resp.) and $l_{n|m}$ be the length of their common border. We define the feature f_b as the maximum ratio between their common border and each of their perimeters:

$$f_b(n, m) = \max\left(\frac{l_{n|m}}{l_n}, \frac{l_{n|m}}{l_m}\right). \quad (5)$$

This cue is most valuable for superpixels that favor color consistency over compactness, such as [13].

3. Size f_s. [27] has empirically shown that size is a powerful cue to prioritize superpixel grouping. Let a_n be the area of the superpixel n as a fraction of the image size. Then the size feature $f_s(n, m)$ is defined as:

$$f_s(n, m) = 1 - a_n - a_m \quad (6)$$

This feature favors the merging of smaller superpixels first.

The resulting weights are the following:

Feature	Color Similarity	Common Border Ratio	Size	Bias
Weight	2.69	1.00	2.36	-3.00

All three cues have similar importance, showing that they all contribute to the probabilities. Note that the scale of the weights and the bias have a major impact on the sigmoid defined in Eq. (2). This highlights the need to learn those weights, as we also show experimentally below (Sec. 5).

4.2. Termination function

Our termination function $\xi(T_k, e_k)$ includes two terms: (1) The probability $(1 - \rho_{n,m})$ that the sampled edge e_k does *not* connect superpixels of the same object. (2) A size term $\alpha(T_k) \in [0, 1]$ computed as the fraction of objects in the training data with area smaller than T_k. In practice, we found the mean of these two terms to give good results:

$$\xi(T_k, e_k) = (1 - \rho_{n,m} + \alpha(T_k))/2. \tag{7}$$

To conclude the implementation description, our code for object proposals using the Randomized Prim's algorithm is available online for download.[2]

5. Experiments

We present in Sec. 5.1 our experimental protocol and the evaluation measures. In Sec. 5.2, we compare several variants of our approach, including colorspaces, a greedy approach and untrained weights. Then, we compare our results to the state of the art in Sec. 5.3.

5.1. Data sets and evaluation protocol

Following previous work [2, 3, 9, 14, 21, 27], we have used the PASCAL VOC 2007 [10] data set to evaluate our approach and compare it to the state of the art. This dataset is composed of $9,963$ images containing objects of 20 different classes. Following [2], we used the classes *bird, car, cat, cow, dog*, and *sheep* for training and the remaining 14 classes for testing, and we removed from the test set the images that had an occurrence of any training class and vice-versa. This left us with $2,941$ images and $9,532$ objects in the test set. [2] had only $7,610$ test objects because they removed those annotated as *difficult* or *truncated*.

For further comparison, we have also evaluated our algorithm against existing methods on the *trainval* set of the larger PASCAL VOC 2012 [11] dataset and test set of the SUN2012 [29] dataset. We followed the same procedure as for VOC 2007, removing from both datasets the images which contained an instance of any of the training classes. This yielded $6,642$ images containing $19,772$ objects for VOC 2012 and $11,811$ images containing $201,353$ objects for the SUN2012. To show the generality of our approach accross datasets, we used the weights learnt on VOC2007.

Our evaluation is based on *IoU*, which measures the quality of a window proposal w with respect to the ground-truth bounding boxes b of objects. Following the definition in [10], $IoU(w, b) = \frac{|w \cap b|}{|w \cup b|}$. The resulting value ranges from 0 (no overlap) to 1 (the windows are the same).

[2]http://www.vision.ee.ethz.ch/software/

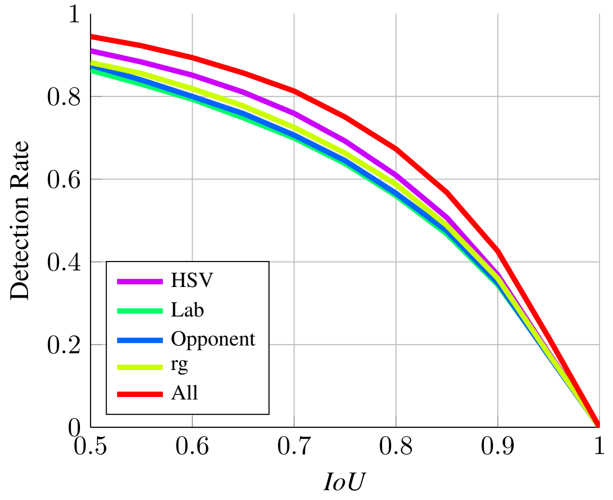


Figure 4: Performance of our algorithm, for $10,000$ proposals, sampling from individual segmentations or from all.

Using a threshold $\theta \in [0.5, 1]$ for the *IoU* as the detection criterion, we measure the *detection rate* as the fraction of objects localized with an *IoU* above θ. This number varies with the *number of proposals* (*#win*) that we ask for each image. In our experiments, we compare detection rates with respect to both θ and *#win* parameters. For brevity, we can resort to the *volume-under-surface* metric (*VUS*), which extends the *area-under-curve* to two parameters. This metric favors, with a single number, methods that can retrieve as many objects, as tightly, and with as few proposals as possible. For computing the *VUS*, we consider linear and logarithmic scales for *#win*. The latter favors high detection rates for low numbers of windows.

5.2. Variants of our approach

In this section, we validate experimentally our model by comparing the following variants:

Individual color spaces. We compare the performance of our algorithm when using the four individual graphs corresponding to different colorspaces, as described in Sec. 4, and sampling among all graphs. As we show in Fig. 4 for 10,000 proposals, the Lab color space gives the worst individual result. Opponent, rg, and HSV sequentially improve the detection rate by up to 5% for $IoU \geq 0.5$. When sampling from the different graphs, we obtain as much as 8% improvement over Lab, and 3% on the best color space (HSV). In terms of linear *VUS*, the combination also improves over individual segmentations: 59% vs. 55% for HSV and rg and 54% for Lab and opponent. This highlights that individual segmentations have less diversity in their proposals than their combination.

Speed. We report in Tab. 1 the decomposition of the time cost of RP when using the fastest (Lab) or all four superpixel segmentations. There is a clear trade-off between detection rate and speed. Using only one segmentation, it takes less than 0.3s to sample 1000 proposals, at the cost of a lower detection rate. When using multiple segmentations, a better detection rate is obtained, but the method is 2.5 times slower. Notably, both variants are still 4 to 10 times faster than the previous fastest method [2] (2.8s), and yield state-of-the-art detection rates.

Edge sampling strategies. To show the benefits of our edge sampling strategy, we compare the following settings: *a*) *Uniform* sampling, discarding weights; *b*) Sampling using weights derived from [27]; *c*) Greedy Prim's algorithm using our learnt weights; *d*) Our proposed RP. For all variants, we used all four segmentation graphs, the same initialization and the same random termination criterion.

In Fig. 5, we show the detection rates at 1000 windows, relative to the performance of uniform sampling. We see that sampling uniformly or with adhoc weights [27] give the worst results. This is because those approaches do not exploit valuable information than can by learnt from training data. Finally, the best performance is obtained when training the combination of features so as to maximize the probability to grow the tree within the same object (Sec. 4.1). The greedy approach obtains higher detection rate for coarse *IoU*, and our RP algorithm outperforms the greedy as well as all other sampling strategies for *IoU*$\geq$ 0.65.

5.3. Comparison with the State-of-the-art

We now compare our method to the state of the art [2, 27, 21] using code available online. To obtain a specific number N of windows from each of them, we use the following procedures. *a*) Selective Search [27] returns window proposals in a single batch of about 2050 windows in average. We uniformly subsample N from this batch; *b*) Objectness [2] provides a score and applies non-maxima suppression (NMS), yielding on average 1850 windows. We keep

Table 1: Split time cost of RP for about 1000 unique proposals, averaged over the 2941 test images of PASCAL VOC2007. [2] and [27] take 2.8s and 3.9s per image (resp.). Sampling is particularly efficient and scales linearly with the number of proposals. The measures have been taken on 1 desktop CPU (3.50GHz). Trivially parallelizable processes are marked with *.

RP	Computational time (s)	
	Lab	All
a) Colorspace conversion	0.06	0.10*
b) Segmentation	0.10	0.40*
c) Feature preprocessing	0.02	0.08*
d) Sampling	0.09*	0.09*
Total	**0.27**	**0.67**
Detection Rate (*IoU* $\geq$0.5)	0.81	0.86

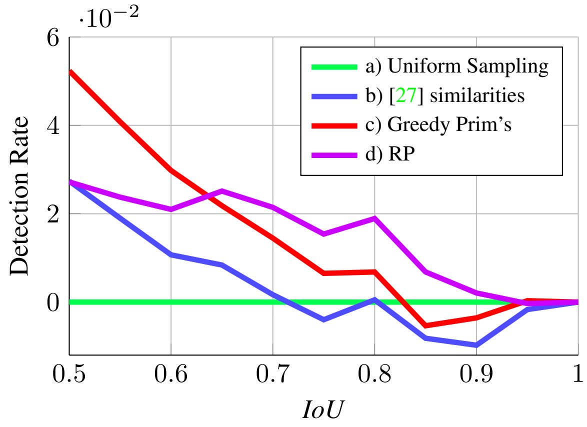


Figure 5: Detection rates for various sampling strategies, relative to the performance of uniform sampling. Learnt weights for random sampling achieves the best performance for *IoU*$\geq$ 0.65.

the N proposals that have the highest scores; *c*) Rahtu [21] proposes 10,000 windows which are re-ranked using NMS. We use the top N ranked windows. *d*) Using our approach, we can directly sample any given number of windows, yet we ensure that we have N unique proposals. Methods that score or rank proposals are expected to obtain much higher detection rates when keeping few ($\leq$100) proposals.

We compare the performance of these methods in terms of the linear/log *VUS* metric in Tab. 2. For the PASCAL VOC 2007 data set RP obtains the best overall *VUS* (0.59/0.28), compared to 0.49/0.25 for the second best method [21, 27]. As expected, the gap is larger for the linear *VUS*. Results are similar for the other datasets (PASCAL VOC 2012 and SUN2012). RP consistently outperforms [2] and [27] on both metrics.

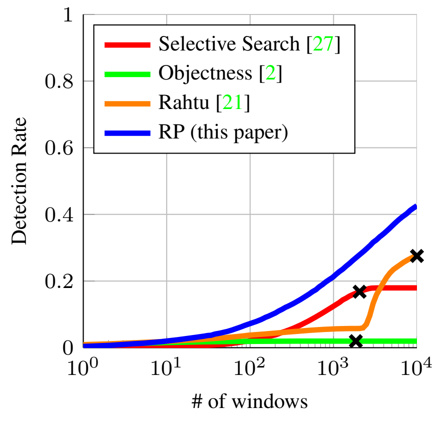
In Fig. 6, we show a more detailed comparison on VOC 2007 for various values of *IoU* threshold and number of object proposals *#win*. We make the following observations.

For thresholds larger than 0.6 (Fig. 6a and 6b), our method outperforms the state-of-the-art scoreless method of [27] for any number of proposed windows, and the gap increases with the threshold. For thresholds $\leq$0.6 (Fig. 6c), [27] is marginally better than RP for 500 to 5000 windows.

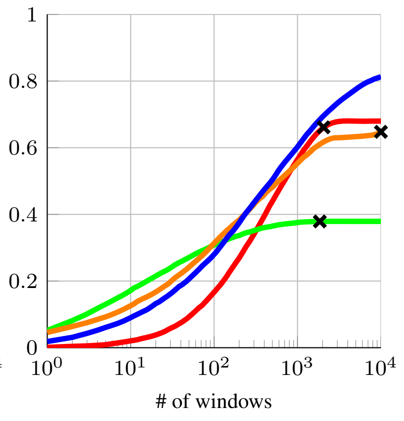
When sampling at least 500 proposals (Fig. 6d and 6e), RP generally also outperforms ranking methods [2, 21]. For few proposals (Fig. 6f), objectness [2] outperforms RP for lower thresholds ($\leq$0.7), then [21] is best between 0.7 and

Table 2: Comparison of VUS with linear or log scaling of *#win*. *#win* varies from 1 to 10,000 proposals, and θ from 0.5 to 1.0.

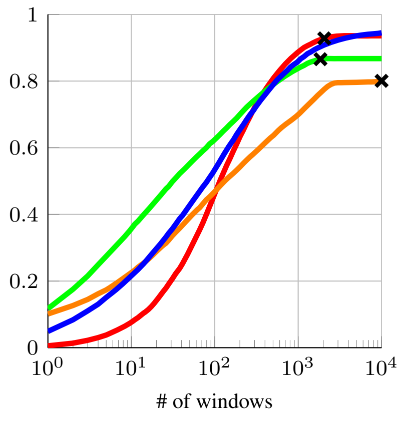
	Volume Under Surface (VUS)					
	VOC 2007		VOC 2012		SUN 2012	
Method	linear	log	linear	log	linear	log
[2]	0.33	0.23	0.33	0.24	0.23	0.15
[21]	0.47	0.25	-	-	-	-
[27]	0.49	0.22	0.52	0.24	0.49	0.23
RP	**0.59**	**0.28**	**0.61**	**0.31**	**0.52**	**0.26**



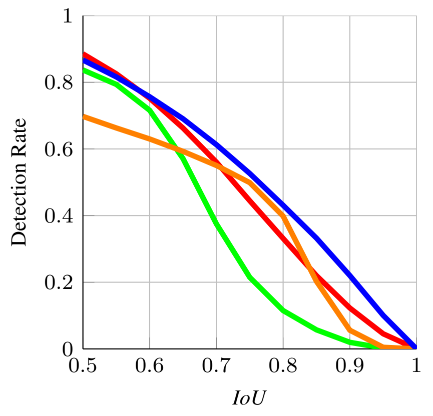
(a) *IoU* ≥ 0.9



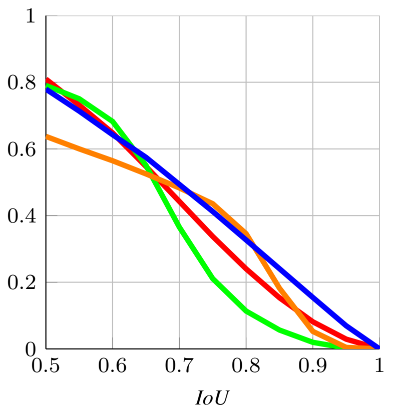
(b) *IoU* ≥ 0.7



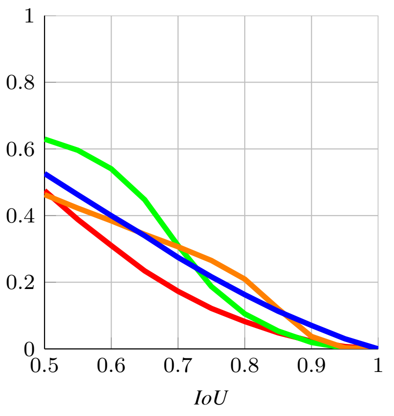
(c) *IoU* ≥ 0.5



(d) 1000 windows



(e) 500 windows



(f) 100 windows

Figure 6: Comparison of detection rates obtained by 3 state-of-the-art methods and ours (RP). We plot the performance for various thresholds of *IoU* (top row) and various numbers of proposal windows (bottom row). Please refer to the text for discussion. The black crosses represent the average number of proposals provided by each method (except ours).

0.85, and RP dominates beyond 0.85. This is because Objectness scores are focused on convex objects, and Rahtu's ranking was designed to boost the recall for higher *IoU* [21]. Both curves show typical signs of using NMS, as the detection rate falls rapidly above a certain *IoU* threshold.

Moreover, note how Objectness fails to find the most difficult objects, as its detection rate is only 37.9% for 1850 windows at 0.7 *IoU*(Fig. 6b), whereas we obtain up to 68.5% detection rate for the same number of windows. On the same setting, [27] obtains 65.2% and [21] 61.1%. The values at 0.9 *IoU* are even more impressive: 2.0% [2], 5.7% [21], 16.4% [27] and 27.9% (RP) for $\#win = 1850$.

In other words, for 1850 proposals, we find 1/4 of the objects perfectly ($IoU \geq 0.9$), 2/3 with very good localization accuracy ($IoU \geq 0.7$), and 90% correctly ($IoU \geq 0.5$). Interestingly, since our method is able to generate many more windows, with 10,000 proposals we can recover more than 40% of the objects perfectly, 80% accurately, and 95% overall. One can observe that apart from the setting where both $\#win \leq 100$ and $IoU \leq 0.65$, our method is either the best performing method or is close to the best method.

We compare the best proposed windows for our approach vs. Objectness in Fig. 1 and provide more examples in Fig. 7 to illustrate the typical accuracy of our detections.

6. Conclusion

In this paper, we have proposed a new method for generic object detection. Contrary to previous work based on grouping superpixels, we have proposed a randomized and very efficient method, which extends Prim's algorithm. Among the benefits, randomization allows our approach to avoid repeating previous mistakes, hence increases the diversity of proposals and the detection rate. As they are bound to superpixel boundaries, the best proposals fit the objects accurately. The performance is improved by maximizing the probability that the sampled groups of superpixels remain within the same object. In the end, our algorithm yields windows significantly faster and significantly more accurately than the state-of-the-art. For instance, 86% of the objects can be found in less than 0.7s using 1000 proposals. In future work, we will develop a score for windows to further increase the detection rate with few proposals.

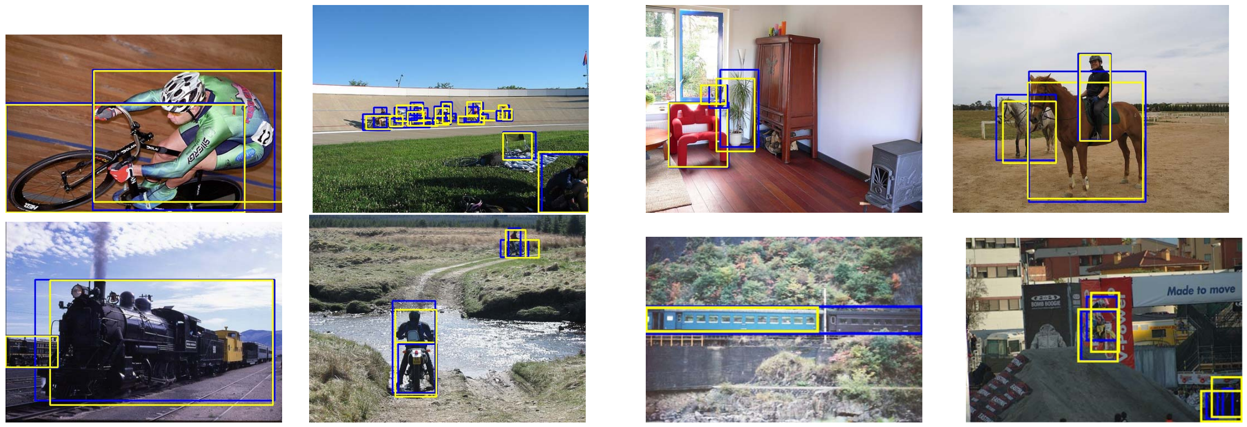


Figure 7: Best windows proposed by our method (yellow), out of 1000, for the ground-truth annotations (blue). The examples in the bottom right show some limitations of our approach. It fails to merge two parts of the same object that are dissimilar and have a thin common border or when the superpixel segmentations miss object boundaries.

Acknowledgments

The authors gratefully acknowledge support by Toyota. This work was supported by the European Research Council (ERC) under the project VarCity (#273940).

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010.

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. on PAMI*, 2012.

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

[4] K.-Y. Chang, T.-L. Liu, H.-T. Chen, and S.-H. Lai. Fusing generic objectness and visual saliency for salient object detection. In *ICCV*, 2011.

[5] R. G. Cinbis and S. Sclaroff. Contextual Object Detection using Set-based Classification. In *ECCV*, 2012.

[6] N. Dalal and B. Triggs. Histogram of Oriented Gradients for human detection. In *CVPR*, 2005.

[7] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *ECCV*, 2010.

[8] T. Deselaers, B. Alexe, and V. Ferrari. Weakly supervised localization and learning with generic knowledge. *IJCV*, 2012.

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

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

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

[13] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 2004.

[14] J. Feng, Y. Wei, L. Tao, C. Zhang, and J. Sun. Salient object detection by composition. In *ICCV*, 2011.

[15] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Trans. on PAMI*, 33(11):2188–2202, 2011.

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

[17] M. Guillaumin and V. Ferrari. Large-scale knowledge transfer for object localization in ImageNet. In *CVPR*, 2012.

[18] Y. J. Lee and K. Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, 2011.

[19] A. Prest, C. Schmid, and V. Ferrari. Weakly supervised learning of interactions between humans and objects. *IEEE Trans. on PAMI*, 2012.

[20] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 1957.

[21] E. Rahtu, J. Kannala, and M. Blaschko. Learning a Category Independent Object Detection Cascade. In *ICCV*, 2011.

[22] F. Sener, C. Bas, and N. Ikizler-Cinbis. On recognizing actions in still images via multiple features. In *ECCV Workshops*, 2012.

[23] P. Siva, C. Russell, and T. Xiang. In defence of negative mining for annotating weakly labelled data. In *ECCV*, 2012.

[24] P. Siva and T. Xiang. Weakly supervised object detector learning with model drift detection. In *ICCV*, 2011.

[25] S. Stalder, H. Grabner, and L. Van Gool. Dynamic Objectness for Adaptive Tracking. In *ACCV*, 2012.

[26] J. Sun and H. Ling. Scale and object aware image retargeting for thumbnail browsing. In *ICCV*, 2011.

[27] K. Van de Sande, J. Uijlings, T. Gevers, and A. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011.

[28] H. Xiao and T. Stibor. Efficient collapsed gibbs sampling for latent dirichlet allocation. In *ACML*, 2010.

[29] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.