BING++: A Fast High Quality Object Proposal Generator at 100fps

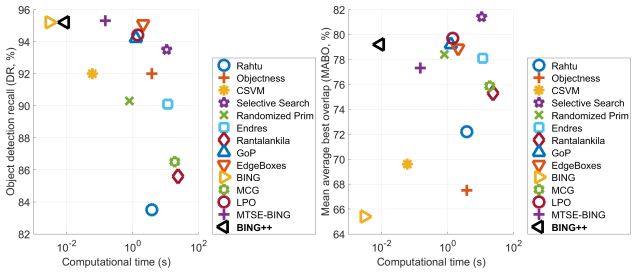
Ziming Zhang†, Yun Liu‡, Tolga Bolukbasi†, Ming-Ming Cheng‡ and Venkatesh Saligrama†
† Department of Electrical & Computer Engineering, Boston University
‡ Department of CCCE & CS, Nankai University, China
{zzhang14, tolgab, srv}@bu.edu nk12csly@mail.nankai.edu.cn cmm@nankai.edu.cn

Abstract

We are motivated by the need for an object proposal generation algorithm that achieves a good balance between proposal localization quality, object recall and computational efficiency. We propose a novel object proposal algorithm BING++ *which inherits the good computational efficiency of BING [7] but significantly improves its proposal localization quality. Central to our success is based on the observation that good bounding boxes are those that tightly cover objects. Edge features, which can be computed efficiently, play a critical role in this context. We propose a new algorithm that recursively improves BING's proposals by exploiting the fact that edges in images are typically associated with object boundaries. BING++ improves proposals recursively by incorporating nearest edge points (to proposal boundary pixels) to obtain a tighter bounding box. This operation has linear computational complexity in number of pixels and can be done efficiently using distance transform. Superpixel merging techniques are then employed as post-processing to further improve the proposal quality. Empirically on the VOC2007 dataset, using* 10^3 *proposals and IoU threshold 0.5, our method achieves 95.3% object detection recall (DR), 79.2% mean average best overlap (MABO), and 68.7% mean average precision (mAP) on object detection over 20 object classes within an average time of* **0.009** *seconds per image.*

1. Introduction

Generic object proposal generation arises as a critical preprocessing step in many applications such as object recognition [32] and detection [14], and consequently has attracted significant attention. Object proposal generation can be broadly measured using three metrics: (a) Object Detection Recall (DR) [7, 36], which is the ratio of the number of correctly detected objects and the total number of objects in the dataset; (b) Proposal Localization Quality measured in terms of average best overlap (ABO) for each object instance in each class, and corresponding mean average best overlap (MABO) across all the classes [29]; (c) Computational Efficiency. In this paper, we are interested in developing new algorithms to provide a small set of windows (*i.e.* bounding boxes) in images with high object detection recall (DR) , high localization quality in terms of mean average best overlap (MABO), and high computational efficiency.

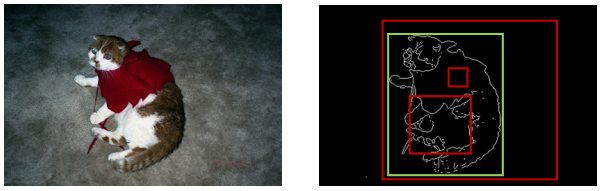


(a) DR vs. computational time (b) MABO vs. computational time

Figure 1. Comparison of state-of-the-art generic object proposal methods on VOC2007 dataset [11] with 10^3 proposals and intersect-over-union (IoU) threshold 0.5. Our method achieves the best trade-off between DR, MABO, and computational time. All the competing results are produced by public codes (see the details in our experimental section).

In recent years while many object proposal generation algorithms have been proposed, existing methods do not appear to achieve a good balance between DR, MABO and computational efficiency. For instance, BING is computationally efficient but has poor localization quality (*e.g.* [7]). Selective Search [29] generates good proposals but is computationally inefficient. Fig. 1 depicts inherent tradeoffs in DR, MABO and computational efficiency. Our perspective is that computational efficiency must be an important consideration since object proposal generation is typically a preprocessing step. Based on this reasoning, we propose a novel object proposal algorithm called *BING++*, which significantly achieves the best trade-off between DR, MABO, and computational time in the literature.

Key insight: We observe that good bounding boxes are those that tightly cover objects and edge features can used to detect object boundaries. In other words we can approximate the boundary of an object by tracing its edges. Edges can be computed efficiently relative to segments and superpixels. Indeed most of the efficient proposal algorithms uti-



(a) original image (b) edge image with bounding boxes

Figure 2. Illustration of the facts that edges can be used to approximate the object (*i.e*. cat) boundary, and the ground-truth bounding box (*i.e*. green one) covers the object as tightly as possible. With the help of edge information, we can push the other 3 predicted bounding boxes (*i.e*. red ones) towards the object boundary by searching for nearest edge points.

lize edge related features. Fig. 2 illustrates our perspective further. As we see, the ground-truth bounding box intersects with the object boundary at a few edge points on each side. In contrast, if the boundary of a proposal does not intersect with any edge point, we can speculate that either this proposal does not cover any object, or it is too loose. Similarly, if the proposal intersects with too many edge points, we expect that it has not yet reached the object boundary. In all of these cases, we would modify the proposals by pushing them towards the object boundaries.

Contributions: In this paper we propose a *fast yet accurate* object proposal algorithm, namely BING++. Our algorithm is based on the BING [7] proposal method by retaining its computational efficiency and good DR but significantly improves its proposal quality. On VOC2007 BING++ achieves 95.3% DR, an MABO of 79.2% with significant improvement of **13.8%** over BING within only **0.009** seconds.

Our goal of seeking tight bounding boxes together with the observation that edges in images are associated with object boundaries suggests a sequential algorithm for updating bounding boxes efficiently. Our algorithm takes BING proposals as input and sequentially improves the bounding boxes. We propose a recursive algorithm that sequentially improves the current bounding box by attempting to incorporate nearest edge points (to boundary pixels) into a new tightest bounding box. This operation has linear computational complexity in number of pixels and can be done efficiently using distance transform. To further improve the proposal quality, we apply fast superpixel merging techniques (*e.g*. [1, 31]) to refine the estimated ground-truth bounding boxes, because superpixels can better locate the object boundaries in images. Then we generate these refined bounding boxes as our final proposals.

1.1. Related Work

The literature of object proposal generation algorithms for images can be categorized into three groups, in general, as follows:

(i) Segmentation/Superpixel based algorithms: In fact most proposal generation algorithms fall into this group. For instance, objectness measure [2] combines saliency, color, edges, and superpixels to score the windows, and then samples bounding boxes with high scores as object proposals. Based on [2], Rahtu *et al*. [27] proposed another cascaded method, where the proposal candidates are sampled from super-pixels based on a prior object localization distribution and then ranked using structured learning with learned features. Further in [5], Blaschko *et al*. investigated the effect of the non-max suppression step in [27] to improve the performance. Uijlings *et al*. [29] proposed selective search by combining the strength of both an exhaustive search and segmentation and being guided by the image structure. Manen *et al*. [24] proposed a randomized Prim algorithm on the superpixel connectivity graphs. Endres and Hoiem [10] proposed ranking the a set of segments using structured learning based on various cues. Krähenbühl and Koltun [19] proposed identifying critical level sets in geodesic distance transforms as proposals, and in [20] they proposed learning ensembles of classifiers for generating proposals. There are several methods based on energy minimization, such as constrained parametric min-cut [6, 18], RIGOR [17], and parametric min-loss [21]. Some other methods utilized segmentation/superpixel grouping using, for instance, segment hierarchy [4, 28, 30, 34] or new distance measure [33]. In general, most of these methods can achieve good localization quality, but suffer from either poor computational efficiency or low DR during testing.

(ii) Edge based algorithms: Compared with segments and superpixels, edges are lightweight visual features in terms of computation. Currently most of the efficient proposal algorithms utilize edge related features. Zhang *et al*. [36] proposed a cascaded ranking SVM (CSVM) method to sample the proposals based on image gradients in a sliding-window manner, and later generalized the method into two-stage cascade SVMs in [35]. Cheng *et al*. [7] proposed the BING algorithm with binary features running at 300fps. Zitnick and Dollár [37] proposed the EdgeBoxes algorithm to fast generate proposals based on edges and contours while achieving good localization quality. Lu *et al*. [23] proposed a contour box algorithm to reject the object proposals without explicit closed contours. In addition, Qi *et al*. [26] proposed a perceptual grouping framework that organizes image edges into meaningful structures, and tested this method for object proposal generation. In general, edge based algorithms are faster than segmentation based algorithm. Among them, BING is the fastest in the literature, but suffers from poor proposal localization quality seriously.

(iii) Proposal post-processing: Several recent works focus on improving proposal quality with small amount of computational cost. For instance, He and Lau [15] proposed an oriented object proposal algorithm for better locating objects by estimating their orientations. Wang *et al*. [31] proposed using multi-thresholding straddling expan-

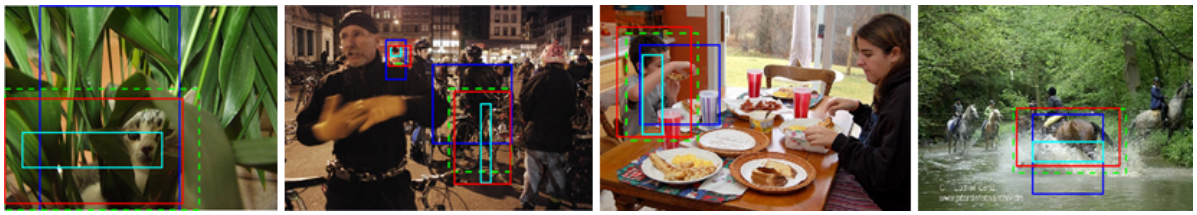


Figure 3. Comparison of best outputs between our edge-based recursive box algorithm (red) and BING (blue). Here dotted green rectangles denote the ground-truth bounding boxes for the objects, and the cyan rectangles denote the initial bounding boxes from BING.

sion (MTSE) to improve quality using superpixels.

Our algorithm BING++ utilizes both edge information and proposal post-processing technique [31] to improve BING, achieving the best trade-off between DR, MABO, and computational time in the literature.

2. Problems in BING

BING [7] is a very efficient object proposal algorithm. Its basic idea is to first train linear filters for each so-called *quantized scale/aspect-ratio* [35, 36] using simple binary gradient features, and then learn another global linear filter to rank bounding boxes and output proposals from the top list. The quantization scheme maps every possible object scale/aspect-ratio to one (or more) of the *predefined and fixed* quantized scales/aspect-ratios. As stated in [35], this quantization scheme reduces the proposal searching space logarithmically, leading to very high computational efficiency. However, this step also leads to significant degradation in proposal quality in practice, as shown in Fig. 3.

To see this, here we show some statistics about the proposal quality on VOC2007 test set. The behavior on training and test sets is similar. We first point to the ABO statistics in Fig. 4. Using percentage of objects as weights, we note that BING's MABO performance is mediocre. We see a clear leftward drift in BING's distribution relative to other methods. This is indicative of the poor proposal localization quality of BING. In order to see how proposals drift from the ground-truth, we point to comparison between the boundary deviation statistics based on percentage of objects and the best proposal deviation from ground-truth bounding box per object in Fig. 5. Ideally a Dirac delta distribution in this context is preferable. Indeed, closer the corresponding distribution is to Dirac delta, the better the proposal algorithm is in terms of localization quality. As we see, compared to other competitors, BING's performance is relatively bad because its distributions appear to have heavier tails on both sides. This indicates that BING is agnostic between choosing larger or smaller proposals for the ground-truth.

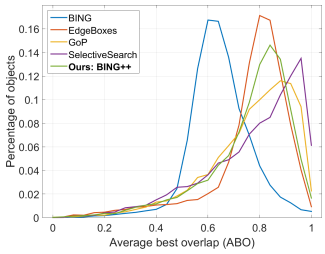


Figure 4. ABO statistic comparison using 10^3 proposals per image and IoU threshold 0.5 on VOC2007 test dataset.

On the other hand since BING has high DR, we conclude that *the quantization scheme in BING leads to poor proposal localization quality*. We infer this based on the view that BING does not allow the proposals to be adaptive to the true object boundaries. In contrast, the methods that fully utilize either edge information (*e.g.* [37]) or segments/superpixels (*e.g.* [29]) work much better and achieve similar performance. By incorporating edge and segmentation information into BING as in our BING++, we expect to see improved performance (see Fig. 4 and 5) without loss in computational speed.

3. Our Solution: BING++

Object proposal generation is about precisely capturing object boundaries, regardless of the pixels inside the objects. Based on this consideration, we propose BING++ to efficiently and effectively approximate ground-truth bounding boxes of objects to capture their boundaries using edges and superpixels in images. It essentially involves three steps: (1) We take proposals from BING as input. (2) We then recursively update current bounding boxes based on their current locations and surrounding edge points to form new bounding boxes. This process is recursed until we find the tightest bounding boxes. (3) We finally apply fast superpixel merging techniques to further refine the output bounding boxes of (2) and output final object proposals.

3.1. Edge-based Recursive Boxes

Overview: The goal of our edge-based recursive box update algorithm is to improve the proposal localization quality while ensuring computational efficiency. Our method works as follows. At time t, we guess an object boundary using the set of nearest edge points to the boundary pixels of current bounding box. If the current bounding box tightly covers the set, we assume that we have reached the ground-truth bounding box. Otherwise, we generate the tightest bounding box to cover those nearest edge points. This new bounding box is proposed as the bounding box for time $t+1$ and serves as an estimate for the ground-truth bounding box. This *deterministic rule* is repeated over time until some termination criterion is satisfied. Fig. 3 illustrates some outputs of our recursive box algorithm.

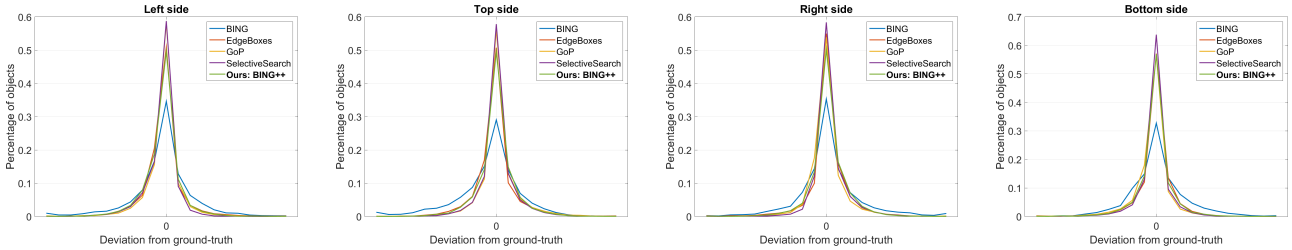
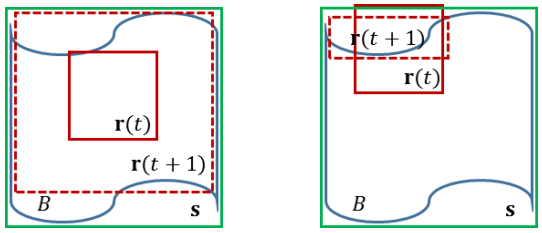


Figure 5. Statistics comparison based on percentage of objects *vs*. best proposal deviation from ground-truth bounding box per object with 10^3 proposals per image and IoU threshold 0.5 on VOC2007 test dataset.



(a) (b)

Figure 6. Illustration of updating current red solid bounding box $\mathbf{r}(t)$ to next red dashed bounding box $\mathbf{r}(t+1)$. Our estimation for $\mathbf{s}$ based on $\mathcal{C}(\mathbf{r}(t))$ succeeds in (a) where the pixels in $\mathcal{C}(\mathbf{r}(t))$ spread well, but fails in (b) where the pixels in $\mathcal{C}(\mathbf{r}(t))$ concentrate on few boundary fragments.

Note that there could be many ways to produce new bounding boxes. For instance, we can update only one side of a bounding box once. In this paper, however, we prefer updating the four sides all together, simply because we want to push the predicted bounding box towards the ground-truth as fast as possible, which may sacrifice some localization quality for speed.

Algorithm: We describe a deterministic algorithm that independently updates bounding boxes. Given an image I, we denote $\mathbf{r}(t) \in \mathbb{R}^4 (t \geq 0)$ as the predicted bounding box at time t, $\mathcal{A}(\mathbf{r}(t)) \subseteq \mathbb{R}^2$ as the set of pixel locations covered by the boundary of $\mathbf{r}(t)$, $\mathcal{B} \subseteq \mathbb{R}^2$ as the *constant* edge map of an image, and $\mathbf{s}$ as the ground-truth bounding box of an object. During training[1] we learn a thresholding parameter and the maximum number of recursive updates that can be used in test time using ground-truth bounding boxes.

Given these inputs our goal is to find a match between $\mathbf{r}(t)$ and $\mathbf{s}$ at some time t^* (*i.e.* $\mathbf{r}(t^*) = \mathbf{s}$). To do so, we first generate a set of pixel locations, $\mathcal{C}(\mathbf{r}(t))$, for $\mathcal{A}(\mathbf{r}(t))$ by looking for the nearest neighbors in $\mathcal{B}$. That is,

$$\mathcal{C}(\mathbf{r}(t)) = \left\{ \mathbf{q} \middle| \arg\min_{\mathbf{q} \in \mathcal{B}} d(\mathbf{p}, \mathbf{q}), \forall \mathbf{p} \in \mathcal{A}(\mathbf{r}(t)) \right\} \subseteq \mathbb{R}^2, \quad (1)$$

where $d(\cdot, \cdot)$ denotes a distance function. For some special distance metric such as Euclidean distance, $\mathcal{C}(\mathbf{r}(t))$ can be efficiently computed using distance transform [12].

Based on this pixel location set $\mathcal{C}(\mathbf{r}(t))$, we define the predicted bounding box, $\mathbf{r}(t+1)$, at time $(t+1)$ as follows:

$$\mathbf{r}(t+1) = \left[\min_{\mathbf{q} \in \mathcal{C}(\mathbf{r}(t))} \mathbf{q}; \max_{\mathbf{q} \in \mathcal{C}(\mathbf{r}(t))} \mathbf{q} \right] \in \mathbb{R}^4, \quad (2)$$

where min and max are entry-wise minimum and maximum operators, and $[\cdot;\cdot]$ denotes the vector concatenating operator. In this way, $\mathbf{r}(t+1)$ is the tightest bounding box which covers all the pixels in set $\mathcal{C}(\mathbf{r}(t))$, and used as an estimate for the location of $\mathbf{s}$. Then the algorithm intuitively works as follows. Whenever $\mathbf{r}(t)$ is indeed approaching $\mathbf{s}$, if $\mathbf{r}(t+1)$ is a good estimator, then we expect that the spatial distortion between $\mathbf{r}(t)$ and $\mathbf{r}(t+1)$ is small. In other words, the overlap score $o(\mathbf{r}(t), \mathbf{r}(t+1))$ between bounding boxes $\mathbf{r}(t)$ and $\mathbf{r}(t+1)$ is large.

Empirically we utilize simple edge detection methods such as canny edge detection to locate the edge points for $\mathcal{B}$ to approximate object boundaries. Many works in the literature (*e.g.* [3, 8, 25]) followed a similar approach. In general accurately detecting object boundaries is also very challenging due to complex imaging factors and semantic ambiguity. Note that better boundary detection such as structured edges [9] may improve the proposal quality at the cost of longer computational time.

Table 1. Performance comparison of different derivatives from BING using VOC2007 test dataset.

Method	DR (%)	MABO (%)	Comp. Time (s)
BING	95.2	65.4	**0.003**
BING + RecursiveBox	91.6	74.3	0.005
BING++	**95.3**	**79.2**	0.009

3.2. Fast Superpixel Merging

Recall that our edge-based recursive box algorithm aims to improve the proposal localization quality very efficiently, on average. As we see in Table 1, the recursive box algorithm does improve the quality in terms of MABO significantly by a large margin of 8.9% only using 0.002 second per image. However, on the other side, it appears to significantly degrade DR down to 91.6% from 95.2%. The reason for this behavior is that actually there is no guarantee that our estimator $\mathbf{r}(t+1)$ will approach the ground-truth bounding box $\mathbf{s}$ eventually. Fig. 6 illustrates two simple cases where our estimation succeeds and fails, respectively.

[1] Essentially this train process may not be necessary based on our experimental results. See Section 3.3 for more discussion.

Algorithm 1 BING++ in testing

Input : an input image I, overlap threshold $\epsilon \geq 0$, number of iterations $T \geq 0$
Output: object proposals Ω

$\mathcal{R} \leftarrow$ BING(I); // object proposals from BING
$\mathcal{B} \leftarrow$ EdgeDetection(I); // edge map for nearest neighbors
$\Omega \leftarrow \emptyset$;
foreach $\mathbf{r}_i(0) \in \mathcal{R}$ **do**
  // edge-based recursive boxes
  for $t = 0$ **to** $T - 1$ **do**
    Calculate $\mathbf{r}_i(t+1)$ based on Eq. 1 and 2 using $\mathbf{r}_i(t)$ and $\mathcal{B}$;
    if $o(\mathbf{r}_i(t), \mathbf{r}_i(t+1)) \geq \epsilon$ **then** break
  end
  // MTSE-based superpixel merging
  $\mathbf{r}_i(t+1) \leftarrow$ MTSE($\mathbf{r}_i(t), I$);
  $\Omega \leftarrow \Omega \bigcup \mathbf{r}_i(t)$; // collecting BING++ proposals
end
return Ω

In (a) the current bounding box $\mathbf{r}(t)$ is surrounded by the edge points in $\mathcal{B}$, implying that the pixels in $\mathcal{C}(\mathbf{r}(t))$ are sufficiently well-spread. This improves the estimate for $\mathbf{s}$. In (b), $\mathbf{r}(t)$ intersects with the edge points. This leads to a situation where $\mathcal{C}(\mathbf{r}(t))$ is determined by a small fraction of $\mathcal{B}$. In practice, there may be scenarios where the correct detections in BING could be updated to wrong bounding boxes.

Therefore, in order to overcome such performance deterioration issues when applying recursive boxes, we propose using the superpixel merging techniques such as [31], namely multi-thresholding straddling expansion (MTSE), as post-processing to improve our proposals. However, this can result in significant computational bottleneck as observed in [31]. The calculation of MTSE takes 0.15 second, making it unsuitable for our purpose. The most time-consuming part in MTSE is the segmentation for generating superpixels[2]. Instead we re-implement MTSE using a GPU version of SLIC [1] to compute superpixels efficiently. In this way we manage to reduce the computational time for MTSE significantly from 0.15 second down to 0.004 second per image using a machine with an Intel Core i7-4790K CPU@4.00GHZ and an NVIDIA GeForce GTX 980.

3.3. BING++

Overall, our proposed BING++ algorithm is essentially a sequential combination of BING, edge-based recursive boxes as one "+", and MTSE-based superpixel merging as the other "+". BING++ retains BINGs DR performance while improving MABO with little degradation in computational time, as shown in Table 1.

We list our BING++ algorithm for object proposal generation in Alg. 1, where $o(\cdot, \cdot)$ denotes the overlap scoring function. In this paper, we utilize the common intersection-over-union (IoU) overlap scoring function defined by the intersection area of two bounding boxes divided by the union of the two boxes. Note that we preserve the same proposal ranking orders from BING based on computational efficiency consideration. The parameters ϵ and T in Alg. 1 are set and fixed as *constants* in our implementation[3] for all the experiments, *i.e.* $\epsilon = 0.95, T = 4$. In such way, our BING++ can be generalized well to generic object proposals. It is also possible to tune these parameters using validation data for different datasets, but we find that in our experiments these two parameters are quite reliable and robust, often achieving good performance.

[2]We downloaded the public code from http://3dimage.ee.tsinghua.edu.cn/cxz/mtse.

Table 2. Detection recall (DR) comparison (%) on VOC2007 test dataset.

Methods	# Proposals, IoU=0.5				# Proposals, IoU=0.7				Time
	1	10	100	1000	1	10	100	1000	(s)
Rahtu [27, 5]	7.0	32.7	64.7	83.5	2.5	15.8	44.7	70.1	3.81
Objectness [2]	17.3	49.5	75.8	92.0	7.4	23.4	37.6	43.1	3.83
CSVM [36]	17.4	33.5	65.1	91.2	5.4	14.8	20.8	27.1	0.06
Sel. Search [29]	9.7	37.3	71.5	93.5	4.1	19.7	49.0	80.0	10.64
Rand. Prim [24]	8.6	35.0	70.4	90.3	3.5	17.3	45.1	73.4	0.79
Endres [10]	**20.9**	**55.2**	**82.8**	90.1	**11.5**	**35.0**	58.0	73.0	11.67
Rantalankila [28]	0.1	0.9	16.2	85.6	0.0	0.4	8.5	67.5	23.72
GoP [19]	2.4	13.8	60.2	94.2	1.3	7.7	35.1	77.8	1.26
EdgeBox [37]	17.8	45.8	75.4	95.1	9.5	30.9	**60.8**	**85.1**	2.07
BING [7]	18.2	37.3	73.0	95.2	7.3	16.9	24.5	29.1	**0.003**
MCG [4]	18.5	44.2	65.7	86.5	9.4	26.9	49.1	70.1	18.97
LPO [20]	18.5	38.0	75.5	94.4	8.0	18.0	49.2	76.8	1.43
MTSE-BING [31]	14.5	37.7	75.2	**95.3**	7.0	18.1	47.2	78.1	0.15
Ours: BING++	16.7	42.2	76.1	**95.3**	8.1	21.1	51.0	81.4	0.009

The computational complexity of our BING++ is linearly proportional to the image size, *i.e.* total number of pixels, roughly speaking. BING has linear complexity due to 2D convolution, the recursive box algorithm is dominated by Euclidean distance transform, which has linear complexity on image size, and MTSE mainly depends on SLIC which also has linear complexity on image size. In order to achieve high computational efficiency for general image sizes as well as preserving good proposal quality, we resize all the images to a fixed size by 375×500 pixels. Empirically we find that this fixed image size is generally good enough for achieving stable and satisfactory performance. Another important reason is to speed up the resource relocation procedure in our GPU implementation for SLIC. This trick leads to nearly constant computational time in practice, *i.e.* 0.009 second per image, at the cost of larger chance of mis-detecting small objects in images with high resolution, because of mis-detection initially by BING.

4. Experiments

We conduct comprehensive experiments to demonstrate that BING++ is extremely efficient as well as capable of generating high quality object proposals and achieving state-of-the-art performance in both object proposal and object detection, respectively.

[3]We will release our code upon acceptance.

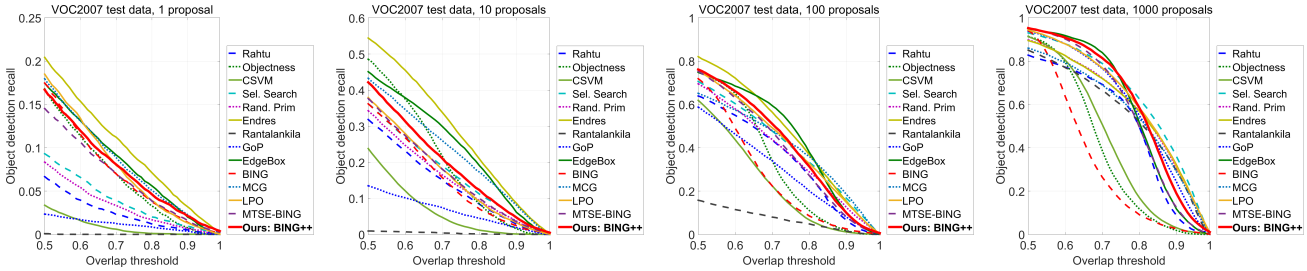


Figure 7. Comparison of recall-overlap curves using different methods and numbers of proposals on VOC2007 test set.

Table 3. ABO & MABO comparison (%) between different proposal algorithms on VOC2007 test dataset using 10^3 proposals with IoU threshold 0.5.

Methods	aer.	bic.	bird	boat	bot.	bus	car	cat	cha.	cow	din.	dog	hor.	mot.	per.	pot.	she.	sofa	tra.	tv	MABO
Rahtu [5, 27]	72.6	74.5	69.0	69.7	49.0	79.9	67.2	82.6	62.9	72.9	80.5	80.6	78.5	74.5	65.7	58.9	69.8	82.3	79.5	73.8	72.2
Objectness [2]	67.3	69.0	65.6	64.4	57.1	72.3	65.3	72.8	64.8	67.6	73.3	71.1	70.9	67.6	63.5	61.4	65.6	74.7	70.6	65.5	67.5
CSVM [36]	70.4	70.2	67.6	66.7	58.9	73.6	67.6	76.5	64.9	68.0	74.1	74.7	72.7	71.4	66.6	62.3	67.9	75.6	74.5	66.7	69.6
Sel. Search [29]	**83.5**	**83.1**	**80.1**	78.1	62.8	85.2	77.7	**90.7**	**77.0**	**83.2**	**88.6**	**89.4**	**82.9**	81.9	72.6	71.2	80.4	**90.3**	**85.7**	**83.7**	**81.4**
Rand. Prim [24]	80.9	80.9	74.5	74.3	59.2	83.7	76.5	87.0	74.6	79.8	87.6	85.1	79.8	81.0	70.6	67.1	72.8	89.4	82.8	80.0	78.4
Endres [10]	71.0	81.0	72.3	65.5	60.9	85.1	79.1	87.9	72.4	80.3	87.0	87.1	82.2	**82.9**	70.5	67.4	76.2	89.7	84.5	79.1	78.1
Rantalankila [28]	73.5	74.6	73.5	66.7	54.0	81.3	72.7	89.2	68.6	76.4	83.2	87.4	81.0	76.0	66.2	62.8	72.1	87.1	82.0	77.5	75.3
GoP [19]	73.8	80.1	76.0	72.2	63.0	**86.0**	**80.3**	88.2	75.9	81.3	85.8	85.7	79.9	79.1	73.5	71.1	78.7	88.4	82.3	82.3	79.2
EdgeBoxes [37]	76.8	81.6	78.5	76.7	**65.8**	83.9	76.8	82.3	76.4	82.2	80.9	83.6	81.3	80.9	**73.6**	**71.7**	**80.8**	82.5	79.8	81.4	78.9
BING [7]	65.5	66.0	64.0	62.3	60.6	66.5	64.4	69.9	62.6	65.1	69.5	68.3	65.9	65.7	63.8	62.4	64.6	69.0	68.6	63.4	65.4
MCG [4]	75.2	77.3	73.3	68.9	55.3	81.4	70.8	87.5	69.6	80.5	82.8	86.0	78.8	75.6	67.9	61.3	78.7	88.7	81.2	76.2	75.9
LPO [20]	74.9	79.9	76.9	72.9	61.4	86.4	80.4	89.1	74.5	82.0	85.1	86.9	82.4	81.7	73.0	71.5	79.4	88.7	85.3	81.6	79.7
MTSE-BING [31]	78.7	77.6	75.5	75.3	63.3	80.6	75.3	83.2	75.8	78.5	82.7	81.9	77.3	78.1	72.1	71.1	76.9	84.0	77.7	79.9	77.3
Ours: BING++	82.9	80.7	77.9	**79.2**	62.2	83.0	77.4	85.7	75.7	81.2	84.0	84.8	80.9	80.5	72.1	70.9	79.7	85.5	79.4	79.2	79.2

We test our method on the PASCAL VOC2007 [11] and Microsoft COCO [22] datasets. VOC2007 contains 20 object categories, and consists of 9963 natural images with object labels and their corresponding ground-truth bounding boxes released for training, validation and test sets. We use the training/validation dataset, consisting of 5011 images, to train BING[4] with its default parameters, and test our BING++ on the test dataset, comprising 4952 images. Microsoft COCO consists of 80 object categories with 82081 images for training and 40137 images for validation, leading to more than 2M annotated instances in total with ground-truth bounding boxes. Besides the amount of images and instances, the contents in images are more complex and challenging than those in VOC2007. On COCO we train our method using training set and test it using validation set.

We measure our performance mainly in terms of (1) object detection recall (DR), (2) average best overlap (ABO) and mean average best overlap (MABO), and (3) computational time. We follow the PASCAL VOC challenge and use IoU overlap threshold 0.5 by default for correct detection.

We compare our method with [2, 4, 5, 10, 24, 19, 20, 27, 28, 29, 37][5], [36][6], [7][7] and [31]. To evaluate the DR and MABO, we download the precomputed proposals for [27][8], [5], [2][9] and [31]. We use the default parameter setting for each method since they have been optimized for VOC2007, in general, expect for [19] where we utilize the parameters $(180, 9)$ as highlighted at the author's website, and for [17] where we set the number of segments to 10^3. We sort all the proposals of different methods based on their predicted scores in a decent order and keep at most top 10^3 proposals for comparison.

4.1. VOC2007

We first compare our BING++ with other proposal algorithms using DR *vs.* IoU overlap threshold in Fig. 7. Overall, our BING++ behaves similarly to many other competitive proposal algorithms such as selective search, edgeBoxes, and GoP, especially when the number of proposals is sufficiently large (*e.g.* 100 and 1000 cases). Note that when the proposals are sufficient, there

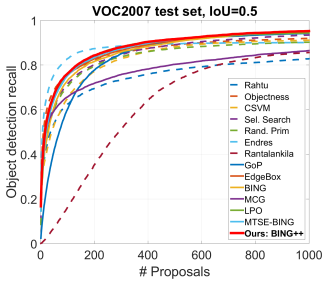


Figure 8. Comparison of detection recall *vs.* number of proposals on VOC2007 test set.

[4] In fact, BING can generalize to generic object proposals without training as shown in [16]. Here we follow orginal BING implementation and train it using train/validation dataset.

[5] For these methods, we use the code at https://github.com/batra-mlp-lab/object-proposals.

[6] https://zimingzhang.wordpress.com/source-code/

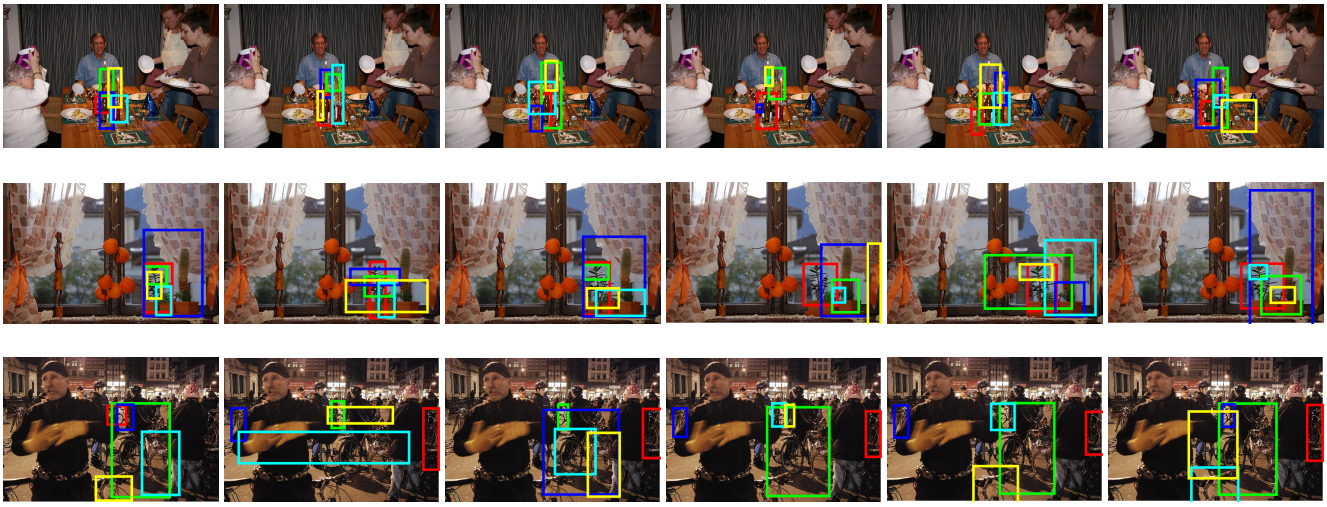
[7] https://github.com/varun-nagaraja/BING-Objectness

[8] http://www.cse.oulu.fi/CMV/Downloads/ObjectDetection/. Same for [5].

[9] http://groups.inf.ed.ac.uk/calvin/objectness/

Table 4. Average precision (AP) & mean AP (mAP) comparison (%) for object detection using different proposal algorithms within fast R-CNN framework on VOC2007 test dataset using 10^3 proposals at most per image and IoU threshold 0.5. Note that our BING++ runs in **0.009s** per image.

Methods (# Proposals)	aer.	bic.	bird	boat	bot.	bus	car	cat	cha.	cow	din.	dog	hor.	mot.	per.	pot.	she.	sofa	tra.	tv	mAP
Rahtu [5, 27] (1,000)	64.3	68.5	59.2	48.6	24.9	79.7	67.9	81.5	29.3	70.5	62.3	76.5	81.6	68.9	58.2	22.5	57.1	62.3	72.1	62.6	60.9
Objectness [2] (1,000)	61.1	64.9	54.0	38.1	25.6	71.8	69.5	75.7	29.7	66.0	57.3	68.2	76.8	61.9	51.4	19.4	51.2	53.9	65.3	59.9	56.1
Sel. Search [29] (999)	71.9	78.3	68.2	53.6	35.4	78.9	75.5	85.6	**42.4**	75.7	70.0	80.5	82.9	72.5	66.3	**35.0**	68.4	66.9	**77.9**	68.4	67.7
Rand. Prim [24] (852)	**76.2**	79.0	63.0	54.8	30.1	80.9	72.2	84.6	39.5	73.5	70.1	77.2	80.6	72.4	64.1	28.3	59.3	69.5	75.8,	66.5	65.9
Endres [10] (830)	66.9	76.2	67.2	47.8	33.4	80.8	75.0	84.3	36.5	71.6	68.0	78.1	77.9	71.7	63.7	27.6	63.7	**69.8**	76.2	67.3	65.2
Rantalankila [28] (856)	68.1	70.2	64.3	49.0	27.2	78.6	69.8	82.4	33.1	71.4	67.0	79.7	78.3	69.1	58.9	26.0	61.9	65.4	76.9	65.8	63.2
GoP [19] (992)	66.8	80.6	**68.6**	50.9	36.4	82.7	**80.6**	85.6	39.3	74.4	**70.9**	82.2	**86.2**	**78.9**	68.2	29.9	66.4	69.5	74.7	**69.7**	68.1
EdgeBoxes [37] (991)	64.2	79.2	66.8	53.6	**38.0**	**83.8**	78.4	84.4	40.1	**78.3**	65.5	81.9	83.3	76.5	**69.6**	34.3	**69.5**	65.1	73.0	67.6	67.7
BING [7] (1,000)	56.7	63.1	55.3	37.4	34.8	70.6	69.9	70.6	27.7	61.7	45.6	63.5	72.5	62.9	57.5	21.8	52.0	46.8	64.9	57.1	54.6
MCG (1,000) [4]	67.0	74.0	64.9	49.9	32.5	77.4	64.9	83.7	34.9	72.1	69.9	78.5	78.0	69.4	60.5	28.8	64.2	67.5	76.4	64.2	63.9
LPO [20] (823)	66.7	**81.9**	68.4	52.5	33.5	81.7	78.4	85.9	40.5	76.4	67.6	81.5	85.7	74.4	68.1	34.7	67.6	66.4	74.6	67.9	67.7
MTSE-BING [31] (996)	70.1	80.6	65.5	54.0	36.7	81.7	79.3	84.6	40.1	78.2	64.2	79.5	84.2	76.8	69.1	33.8	67.2	67.5	73.5	67.9	67.7
Ours: BING++ (1,000)	72.8	78.4	68.4	**57.0**	**38.0**	83.0	79.6	**86.8**	41.5	76.9	68.1	**82.3**	83.8	77.6	68.1	32.5	66.8	68.4	74.9	68.4	**68.7**



(a) Ours: BING++ (b) BING (c) EdgeBoxes (d) LPO (e) MTSE-BING (f) Selective Search

Figure 9. Illustration of detection results on VOC2007 test dataset for some comparative proposal algorithms using classes of (top) bottle, (middle) potted plants, and (bottom) bicycle. In each image, we show top-5 detections for the class, colored by red, green, blue, cyan, and yellow.

are big gaps between BING and our BING++, indicating that BING++ achieves huge improvement over BING.

To quantify these plots, we list the corresponding numbers as well the average computational time of each method in Table 2. In both cases of IoU threshold equal to 0.5 or 0.7, BING++ can always achieve similar performance to the best ones. However, it is quite notable that BING++ is much faster than other competitive proposal algorithms. For instance, compared BING++ with selective search, BING++ is 10^3 times faster. Also by comparing BING++ with BING, we see that with IoU threshold equal to 0.5 using only 1 proposal BING++ is worse than BING. This is probably because of the performance deterioration issue in the recursive box algorithm. By checking IoU threshold equal to 0.7 with 1000 proposals, there is a huge improvement again from 29.1% jumping to 81.4%. This well demonstrates the capability of BING++ for high quality object proposals.

Next we compare different algorithms using DR *vs*. number of proposals and show the results in Fig. 8(a). Clearly with IoU threshold equal to 0.5, BING++ performs among the top, which is consistent with Table 2.

Table 3 list our comparison on ABO and MABO. Still BING++ performs consistently close to the best performance among the competitors and finally achieves 79.2% MABO, only 2.2% smaller than selective search. As shown in [37], considering overall achievement this small difference is negligible in terms of proposal quality.

To see this point, we conduct the object detection task to measure the impact of DR, ABO and MABO of proposals on real applications, and list the results in Table 4. We run different algorithms to generate proposals and feed them to fast R-CNN [13] to perform detection, respectively. In terms of mAP, the overall detection performance of each competitive proposal algorithm is quite close to each other, *i.e*. selective search, GoP, EdgeBoxes, LPO, MTSE-BING, and BING++, and our BING++ is slightly better. Also the AP for each class from BING++ is close to the best performance among the competitors. We emphasize that the

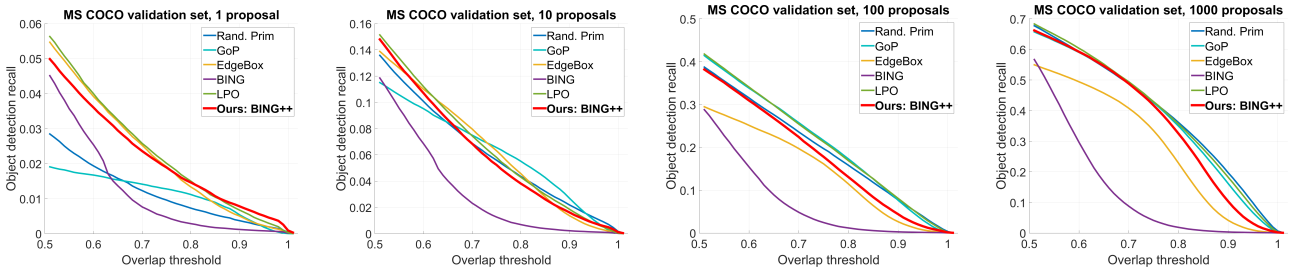


Figure 10. Comparison of recall-overlap curves using different methods and numbers of proposals on MS COCO validation set.

Table 5. Detection recall (DR) comparison (%) on COCO validation set.

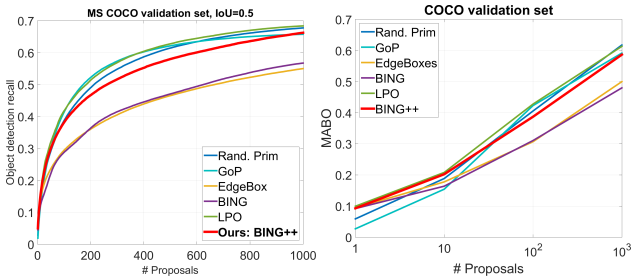
Methods	# Proposals, IoU=0.5				# Proposals, IoU=0.7			
	1	10	100	1000	1	10	100	1000
Rand. Prim [24]	2.9	13.6	38.7	67.7	1.2	6.6	22.8	47.7
GoP [19]	1.9	11.5	41.4	65.7	1.4	7.3	**24.9**	47.8
EdgeBox [37]	5.5	13.9	29.5	55.0	2.4	**7.6**	19.0	39.6
BING [7]	4.5	11.9	28.8	56.7	0.7	2.1	4.3	7.6
LPO [20]	**5.6**	**15.1**	**41.8**	**68.4**	**2.5**	7.1	24.6	**48.2**
Ours: BING++	5.0	14.8	38.2	66.2	2.3	6.5	21.4	47.5

running time of BING++ is only 0.009 second per image, which is much faster than any of the existing competitive proposal algorithms with good quality.

To better view the detection difference between different proposal algorithms, we illustrate some results in Fig. 9. Compared with BING, our BING++ produces more reasonable detections. Interestingly, the attentions of all the comparative algorithms are more or less the same even in such complex images, while the predicted bounding boxes vary.

4.2. Microsoft COCO

As on VOC2007, we first compare different algorithms using DR *vs*. IoU overlap threshold in Fig. 10. Due to the large size of the dataset, here we only compare several relatively efficient algorithms. Again BING++ performs reasonably well among the top for all the cases. Interestingly EdgeBoxes seems to struggle on COCO. One possible reason is that images in COCO are more complex than those in VOC2007, in general, leading to more edges which confuse EdgeBoxes. Another possible reason is that EdgeBoxes is quite sensitive to its parameters, and we need to re-tune its parameters using training data in COCO. However, our BING++ is more robust to parameter settings. We list in Table 5 the corresponding numbers in Fig. 10 for numerical comparison. Also in Fig. 11(a) we show the behavior of different algorithms using DR *vs*. number of proposals. BING++ performs slightly worse. We think that this occurs probably because of the image resizing for computational efficiency in BING++. To see the proposal localization quality we show the comparison of MABO *vs*. number of proposals in Fig. 11(b). Similarly, BING++ performs slightly worse especially when number of proposals is larger. Note that compared with BING, the DR of BING++ is significantly higher from 7.6% jumping to



(a) (b)

Figure 11. Comparison of **(a)** detection recall *vs*. number of proposals, and **(b)** MABO *vs*. number of proposals, on MS COCO validation set.

47.5% with IoU threshold equal to 0.7.

Note that on COCO our BING++ still runs at 0.009 second per image, while others run even slower than the time listed in Table 2, because the average image size in COCO is larger than that in VOC2007.

5. Conclusion

In this paper, we propose a novel object proposal algorithm, BING++, for generating high quality proposals in a very efficient way. Our algorithm essentially improves the proposal quality of BING significantly as well as preserves its high computational efficiency. BING++ consists of three components sequentially: (1) BING algorithm, (2) our new proposed edge-based recursive box algorithm, (3) a re-implemented MTSE superpixel merging algorithm as post-processing. We leverage the facts that edges in images can be used to approximate object boundaries, and the ground-truth bounding boxes should cover the entire objects as tightly as possible. Based on these considerations, we propose estimating ground-truth bounding boxes as a recursive system, where in each iteration, a new bounding box is generated by covering the set of nearest edge points to the boundary pixels of current bounding box as tightly as possible. Using distance transform, we can operate this algorithm in roughly linear time in terms of number of pixels. A GPU version of MTSE algorithm is utilized to refine the proposal quality. Comprehensive experiments are conducted on VOC2007 and Microsoft COCO. Our BING++ achieves the state-of-the-art in terms of not only proposal quality but also running speed on average.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012. 2, 5

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 2012. 2, 5, 6, 7

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

[4] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 2, 5, 6, 7

[5] M. B. Blaschko, J. Kannala, and E. Rahtu. Non maximal suppression in cascaded ranking models. In J.-K. Kämäräinen and M. Koskela, editors, *Image Analysis*, volume 7944 of *Lecture Notes in Computer Science*, pages 408–419. Springer, 2013. 2, 5, 6, 7

[6] J. Carreira and C. Sminchisescu. CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts. *IEEE TPAMI*, 34(7):1312–1328, 2012. 2

[7] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, 2014. 1, 2, 3, 5, 6, 7, 8

[8] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, volume 2, pages 1964–1971, 2006. 4

[9] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1841–1848. IEEE, 2013. 4

[10] I. Endres and D. Hoiem. Category-independent object proposals with diverse ranking. *IEEE TPAMI*, 36(2):222–234, 2014. 2, 5, 6, 7

[11] 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. 1, 6

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8(1):415–428, 2012. 4

[13] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 7

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587. IEEE, 2014. 1

[15] S. He and R. W. Lau. Oriented object proposals. In *ICCV*, 2015. 2

[16] J. Hosang, R. Benenson, and B. Schiele. How good are detection proposals, really? In *BMVC*, 2014. 6

[17] A. Humayun, F. Li, and J. M. Rehg. RIGOR: Reusing Inference in Graph Cuts for generating Object Regions. In *CVPR*, june 2014. 2, 6

[18] A. Humayun, F. Li, and J. M. Rehg. The middle child problem: Revisiting parametric min-cut and seeds for object proposals. In *ICCV*, 2015. 2

[19] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*, pages 725–739, 2014. 2, 5, 6, 7, 8

[20] P. Krähenbühl and V. Koltun. Learning to propose objects. In *CVPR*, 2015. 2, 5, 6, 7, 8

[21] T. Lee, S. Fidler, and S. Dickinson. Learning to combine mid-level cues for object proposal generation. In *ICCV*, 2015. 2

[22] 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, 2014. 6

[23] C. Lu, S. Liu, J. Jia, and C.-K. Tang. Contour box: Rejecting object proposals without explicit closed contours. In *ICCV*, 2015. 2

[24] S. Manén, M. Guillaumin, and L. Van Gool. Prime Object Proposals with Randomized Prim's Algorithm. In *ICCV*, Dec. 2013. 2, 5, 6, 7, 8

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

[26] Y. Qi, Y.-Z. Song, T. Xiang, H. Zhang, T. Hospedales, Y. Li, and J. Guo. Making better use of edges via perceptual grouping. In *CVPR*, 2015. 2

[27] E. Rahtu, J. Kannala, and M. Blaschko. Learning a category independent object detection cascade. In *ICCV*, 2011. 2, 5, 6, 7

[28] P. Rantalankila, J. Kannala, and E. Rahtu. Generating object segmentation proposals using global and local search. In *CVPR*, pages 2417–2424, 2014. 2, 5, 6, 7

[29] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013. 1, 2, 3, 5, 6, 7

[30] C. Wang, L. Zhao, S. Liang, L. Zhang, J. Jia, and Y. Wei. Object proposal by multi-branch hierarchical segmentation. 2

[31] X. C. H. M. X. Wang and Z. Zhao. Improving object proposals with multi-thresholding straddling expansion. In *CVPR*, 2015. 2, 3, 5, 6, 7

[32] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. Cnn: Single-label to multi-label. *arXiv preprint arXiv:1406.5726*, 2014. 1

[33] Y. Xiao, C. Lu, E. Tsougenis, Y. Lu, and C.-K. Tang. Complexity-adaptive distance metric for object proposals generation. In *CVPR*, 2015. 2

[34] V. Yanulevskaya, J. Uijlings, and N. Sebe. Learning to group objects. In *CVPR*, 2014. 2

[35] Z. Zhang and P. H. S. Torr. Object proposal generation using two-stage cascade svms. *CoRR*, abs/1407.5242, 2014. 2, 3

[36] Z. Zhang, J. Warrell, and P. H. Torr. Proposal generation for object detection using cascaded ranking svms. In *CVPR*, 2011. 1, 2, 3, 5, 6

[37] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 2, 3, 5, 6, 7, 8