Edge Boxes: Locating Object Proposals from Edges

C. Lawrence Zitnick and Piotr Dollár

Microsoft Research

Abstract. The use of object proposals is an effective recent approach for increasing the computational efficiency of object detection. We propose a novel method for generating object bounding box proposals using edges. Edges provide a sparse yet informative representation of an image. Our main observation is that the number of contours that are wholly contained in a bounding box is indicative of the likelihood of the box containing an object. We propose a simple box objectness score that measures the number of edges that exist in the box minus those that are members of contours that overlap the box's boundary. Using efficient data structures, millions of candidate boxes can be evaluated in a fraction of a second, returning a ranked set of a few thousand top-scoring proposals. Using standard metrics, we show results that are significantly more accurate than the current state-of-the-art while being faster to compute. In particular, given just 1000 proposals we achieve over 96% object recall at overlap threshold of 0.5 and over 75% recall at the more challenging overlap of 0.7. Our approach runs in 0.25 seconds and we additionally demonstrate a near real-time variant with only minor loss in accuracy.

Keywords: object proposals, object detection, edge detection

1 Introduction

The goal of object detection is to determine whether an object exists in an image, and if so where in the image it occurs. The dominant approach to this problem over the past decade has been the sliding windows paradigm in which object classification is performed at every location and scale in an image [1–3]. Recently, an alternative framework for object detection has been proposed. Instead of searching for an object at every image location and scale, a set of object bounding box proposals is first generated with the goal of reducing the set of positions that need to be further analyzed. The remarkable discovery made by these approaches [4–11] is that object proposals may be accurately generated in a manner that is agnostic to the type of object being detected. Object proposal generators are currently used by several state-of-the-art object detection algorithms [5, 12, 13], which include the winners of the 2013 ImageNet detection challenge [14] and top methods on the PASCAL VOC dataset [15].

High recall and *efficiency* are critical properties of an object proposal generator. If a proposal is not generated in the vicinity of an object that object

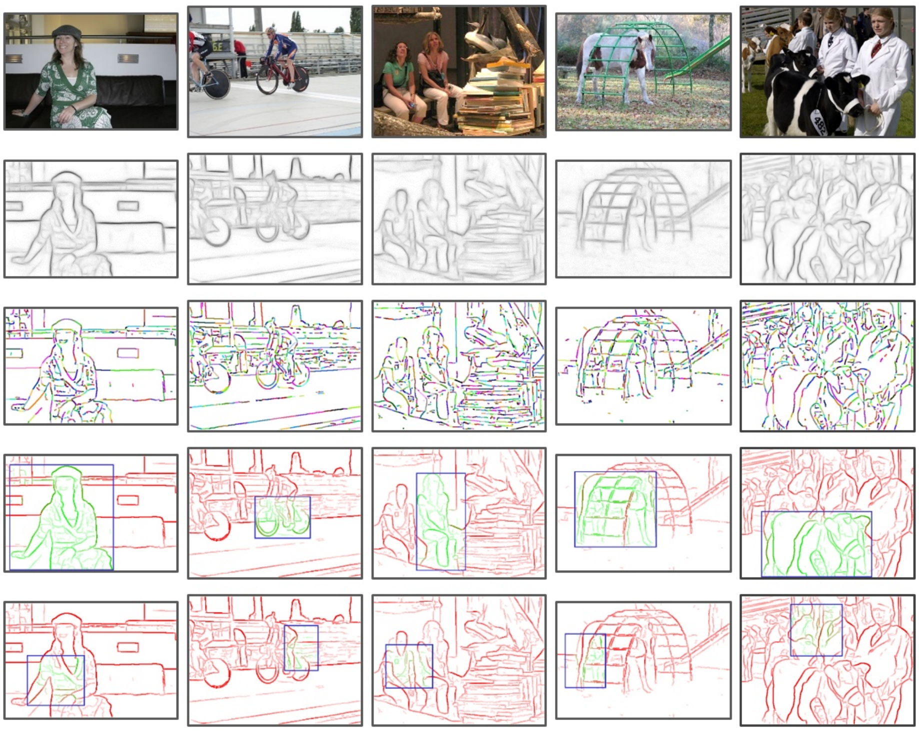

Fig. 1. Illustrative examples showing from top to bottom (first row) original image, (second row) Structured Edges [16], (third row) edge groups, (fourth row) example correct bounding box and edge labeling, and (fifth row) example incorrect boxes and edge labeling. Green edges are predicted to be part of the object in the box ($w_b(s_i) = 1$), while red edges are not ($w_b(s_i) = 0$). Scoring a candidate box based solely on the number of contours it *wholly encloses* creates a surprisingly effective object proposal measure. The edges in rows 3-5 are thresholded and widened to increase visibility.

cannot be detected. An effective generator is able to obtain high recall using a relatively modest number of candidate bounding boxes, typically ranging in the hundreds to low thousands per image. The precision of a proposal generator is less critical since the number of generated proposals is a small percentage of the total candidates typically considered by sliding window approaches (which may evaluate tens to hundreds of thousands of locations *per object category*). Since object proposal generators are primarily used to reduce the computational cost of the detector, they should be significantly faster than the detector itself. There is some speculation that the use of a small number of object proposals may even improve detection accuracy due to reduction of spurious false positives [4].

In this paper we propose *Edge Boxes*, a novel approach to generating object bounding box proposals directly from edges. Similar to segments, edges provide a simplified but informative representation of an image. In fact, line drawings of an image can accurately convey the high-level information contained in an image using only a small fraction of the information [17, 18]. As we demonstrate, the use of edges offers many computational advantages since they may be efficiently computed [16] and the resulting edge maps are sparse. In this work we investigate how to directly detect object proposals from edge-maps.

Our main contribution is the following observation: the number of contours *wholly enclosed* by a bounding box is indicative of the likelihood of the box containing an object. We say a contour is wholly enclosed by a box if all edge pixels belonging to the contour lie within the interior of the box. Edges tend to correspond to object boundaries, and as such boxes that tightly enclose a set of edges are likely to contain an object. However, some edges that lie within an object's bounding box may not be part of the contained object. Specifically, edge pixels that belong to contours straddling the box's boundaries are likely to correspond to objects or structures that lie outside the box, see Figure 1. In this paper we demonstrate that scoring a box based on the number of contours it *wholly encloses* creates a surprisingly effective proposal measure. In contrast, simply counting the number of edge pixels within the box is not as informative. Our approach bears some resemblance to superpixels straddling measure introduced by [4]; however, rather than measuring the number of straddling contours we instead remove such contours from consideration.

As the number of possible bounding boxes in an image is large, we must be able to score candidates efficiently. We utilize the fast and publicly available Structured Edge detector recently proposed in [16, 19] to obtain the initial edge map. To aid in later computations, neighboring edge pixels of similar orientation are clustered together to form groups. Affinities are computed between edge groups based on their relative positions and orientations such that groups forming long continuous contours have high affinity. The score for a box is computed by summing the edge strength of all edge groups within the box, minus the strength of edge groups that are part of a contour that straddles the box's boundary, see Figure 1.

We evaluate candidate boxes utilizing a sliding window approach, similar to traditional object detection. At every potential object position, scale and aspect ratio we generate a score indicating the likelihood of an object being present. Promising candidate boxes are further refined using a simple coarse-to-fine search. Utilizing efficient data structures, our approach is capable of rapidly finding the top object proposals from among millions of potential candidates.

We show improved recall rates over state-of-the-art methods for a wide range of intersection over union thresholds, while simultaneously improving efficiency. In particular, on the PASCAL VOC dataset [15], given just 1000 proposals we achieve over 96% object recall at overlap threshold of 0.5 and over 75% recall at an overlap of 0.7. At the latter and more challenging setting, previous state-of-the-art approaches required considerably more proposals to achieve similar recall. Our approach runs in quarter of a second, while a near real-time variant runs in a tenth of a second with only a minor loss in accuracy.

2 Related work

The goal of generating object proposals is to create a relatively small set of candidate bounding boxes that cover the objects in the image. The most common use of the proposals is to allow for efficient object detection with complex and expensive classifiers [5, 12, 13]. Another popular use is for weakly supervised learning [20, 21], where by limiting the number of candidate regions, learning with less supervision becomes feasible. For detection, recall is critical and thousands of candidates can be used, for weakly supervised learning typically a few hundred proposals per image are kept. Since it's inception a few years ago [4, 9, 6], object proposal generation has found wide applicability.

Generating object proposals aims to achieve many of the benefits of image segmentation without having to solve the harder problem of explicitly partitioning an image into non-overlapping regions. While segmentation has found limited success in object detection [22], in general it fails to provide accurate object regions. Hoiem et al. [23] proposed to use multiple overlapping segmentations to overcome errors of individual segmentations, this was explored further by [24] and [25] in the context of object detection. While use of multiple segmentations improves robustness, constructing coherent segmentations is an inherently difficult task. Object proposal generation seeks to sidestep the challenges of full segmentation by directly generating multiple overlapping object proposals.

Three distinct paradigms have emerged for object proposal generation. Candidate bounding boxes representing object proposals can be found by measuring their 'objectness' [4, 11], producing multiple foreground-background segmentations of an image [6, 9, 10], or by merging superpixels [5, 8]. Our approach provides an alternate framework based on *edges* that is both simpler and more efficient while sharing many advantages with previous work. Below we briefly outline representative work for each paradigm; we refer readers to Hosang et al. [26] for a thorough survey and evaluation of object proposal methods.

Objectness Scoring: Alexe et al. [4] proposed to rank candidates by combining a number of cues in a classification framework and assigning a resulting 'objectness' score to each proposal. [7] built on this idea by learning efficient cascades to more quickly and accurately rank candidates. Among multiple cues, both [4] and [7] define scores based on edge distributions near window boundaries. However, these edge scores do not remove edges belonging to contours intersecting the box boundary, which we found to be critical. [4] utilizes a superpixel straddling measure penalizing candidates containing segments overlapping the boundary. In contrast, we suppress straddling contours by propagating information across edge groups that may not directly lie on the boundary. Finally, recently [11] proposed a very fast objectness score based on image gradients.

Seed Segmentation: [6, 9, 10] all start with multiple seed regions and generate a separate foreground-background segmentation for each seed. The primary advantage of these approaches is generation of high quality segmentation masks, the disadvantage is their high computation cost (minutes per image).

Superpixel Merging: Selective Search [5] is based on computing multiple hierarchical segmentations based on superpixels from [27] and placing bounding boxes around them. Selective Search has been widely used by recent top detection methods [5, 12, 13] and the key to its success is relatively fast speed (seconds per image) and high recall. In a similar vein, [8] propose a randomized greedy algorithm for computing sets of superpixels that are likely to occur together. In our work, we operate on groups of edges as opposed to superpixels. Edges can be represented probabilistically, have associated orientation information, and can be linked allowing for propagation of information; properly exploited, this additional information can be used to achieve large gains in accuracy.

As far as we know, our approach is the first to generate object bounding box proposals directly from edges. Unlike all previous approaches we do not use segmentations or superpixels, nor do we require learning a scoring function from multiple cues. Instead we propose to score candidate boxes based on the number of contours *wholly enclosed* by a bounding box. Surprisingly, this conceptually simple approach out-competes previous methods by a significant margin.

3 Approach

In this section we describe our approach to finding object proposals. Object proposals are ranked based on a single score computed from the contours wholly enclosed in a candidate bounding box. We begin by describing a data structure based on edge groups that allows for efficient separation of contours that are fully enclosed by the box from those that are not. Next, we define our edge-based scoring function. Finally, we detail our approach for finding top-ranked object proposals that uses a sliding window framework evaluated across position, scale and aspect ratio, followed by refinement using a simple coarse-to-fine search.

Given an image, we initially compute an edge response for each pixel. The edge responses are found using the Structured Edge detector [16, 19] that has shown good performance in predicting object boundaries, while simultaneously being very efficient. We utilize the single-scale variant with the sharpening enhancement introduced in [19] to reduce runtime. Given the dense edge responses, we perform Non-Maximal Suppression (NMS) orthogonal to the edge response to find edge peaks, Figure 1. The result is a sparse edge map, with each pixel p having an edge magnitude m_p and orientation θ_p. We define edges as pixels with $m_p > 0.1$ (we threshold the edges for computational efficiency). A contour is defined as a set of edges forming a coherent boundary, curve or line.

3.1 Edge groups and affinities

As illustrated in Figure 1, our goal is to identify contours that overlap the bounding box boundary and are therefore unlikely to belong to an object contained by the bounding box. Given a box b, we identify these edges by computing for each $p \in b$ with $m_p > 0.1$ its maximum affinity with an edge on the box boundary. Intuitively, edges connected by straight contours should have high affinity, where those not connected or connected by a contour with high curvature should have lower affinity. For computational efficiency we found it advantageous to group edges that have high affinity and only compute affinities between edge groups. We form the edge groups using a simple greedy approach that combines 8-connected edges until the sum of their orientation differences is above a threshold ($\pi/2$). Small groups are merged with neighboring groups. An illustration of the edge groups is shown in Figure 1, row 3.

Given a set of edge groups $s_i \in S$, we compute an affinity between each pair of neighboring groups. For a pair of groups s_i and s_j, the affinity is computed based on their mean positions x_i and x_j and mean orientations θ_i and θ_j. Intuitively, edge groups have high affinity if the angle between the groups' means in similar to the groups' orientations. Specifically, we compute the affinity $a(s_i, s_j)$ using:

$$a(s_i, s_j) = |\cos(\theta_i - \theta_{ij}) \cos(\theta_j - \theta_{ij})|^{\gamma}, \tag{1}$$

where θ_{ij} is the angle between x_i and x_j. The value of γ may be used to adjust the affinity's sensitivity to changes in orientation, with $\gamma = 2$ used in practice. If two edge groups are separated by more than 2 pixels their affinity is set to zero. For increased computational efficiency only affinities above a small threshold (0.05) are stored and the rest are assumed to be zero.

The edge grouping and affinity measure are computationally trivial. In practice results are robust to the details of the edge grouping.

3.2 Bounding box scoring

Given the set of edge groups S and their affinities, we can compute an object proposal score for any candidate bounding box b. To find our score, we first compute the sum m_i of the magnitudes m_p for all edges p in the group s_i. We also pick an arbitrary pixel position $\bar{x}_i$ of some pixel p in each group s_i. As we will show, the exact choice of $p \in s_i$ does not matter.

For each group s_i we compute a continuous value $w_b(s_i) \in [0, 1]$ that indicates whether s_i is wholly contained in b, $w_b(s_i) = 1$, or not, $w_b(s_i) = 0$. Let S_b be the set of edge groups that overlap the box b's boundary. We find S_b using an efficient data structure that is described in Section 3.3. For all $s_i \in S_b$, $w_b(s_i)$ is set to 0. Similarly $w_b(s_i) = 0$ for all s_i for which $\bar{x}_i \notin b$, since all of its pixels must either be outside of b or $s_i \in S_b$. For the remaining edge groups for which $\bar{x}_i \in b$ and $s_i \notin S_b$ we compute $w_b(s_i)$ as follows:

$$w_b(s_i) = 1 - \max_{T} \prod_{j}^{|T|-1} a(t_j, t_{j+1}), \tag{2}$$

where T is an ordered path of edge groups with a length of $|T|$ that begins with some $t_1 \in S_b$ and ends at $t_{|T|} = s_i$. If no such path exists we define $w_b(s_i) = 1$. Thus, Equation (2) finds the path with highest affinity between the edge group s_i and an edge group that overlaps the box's boundary. Since most pairwise affinities are zero, this can be done efficiently.

Using the computed values of w_b we define our score using:

$$h_b = \frac{\sum_i w_b(s_i) m_i}{2(b_w + b_h)^{\kappa}}, \tag{3}$$

where b_w and b_w are the box's width and height. Note that we divide by the box's perimeter and not its area, since edges have a width of one pixel regardless of scale. Nevertheless, a value of $\kappa = 1.5$ is used to offset the bias of larger windows having more edges on average.

In practice we use an integral image to speed computation of the numerator in Equation (3). The integral image is used to compute the sum of all m_i for which $\bar{x}_i \in b$. Next, for all s_i with $\bar{x}_i \in b$ and $w_b(s_i) < 1$, $(1 - w_b(s_i))m_i$ is subtracted from this sum. This speeds up computation considerably as typically $w_b(s_i) = 1$ for most s_i and all such s_i do not need to be explicitly considered.

Finally, it has been observed that the edges in the center of the box are of less importance than those near the box's edges [4]. To account for this observation we can subtract the edge magnitudes from a box b^{in} centered in b:

$$h_b^{in} = h_b - \frac{\sum_{p \in b^{in}} m_p}{2(b_w + b_h)^\kappa}, \tag{4}$$

where the width and height of b^{in} is $b_w/2$ and $b_h/2$ respectively. The sum of the edge magnitudes in b^{in} can be efficiently computed using an integral image. As shown in Section 4 we found h_b^{in} offers slightly better accuracy than h_b with minimal additional computational cost.

3.3 Finding intersecting edge groups

In the previous section we assumed the set of edge groups S_b that overlap the box b's boundary is known. Since we evaluate a huge number of bounding boxes (Sec. 3.4), an efficient method for finding S_b is critical. Naive approaches such as exhaustively searching all of the pixels on the boundary of a box would be prohibitively expensive, especially for large boxes.

We propose an efficient method for finding intersecting edge groups for each side of a bounding box that relies on two additional data structures. Below, we describe the process for finding intersections along a horizontal boundary from pixel (c_0, r) to (c_1, r). The vertical boundaries may be handled in a similar manner. For horizontal boundaries we create two data structures for each row of the image. The first data structure stores an ordered list L_r of edge group indices for row r. The list is created by storing the order in which the edge groups occur along the row r. An index is only added to L_r if the edge group index changes from one pixel to the next. The result is the size of L_r is much smaller than the width of the image. If there are pixels between the edge groups that are not edges, a zero is added to the list. A second data structure K_r with the same size as the width of the image is created that stores the corresponding index into L_r for each column c in row r. Thus, if pixel p at location (c, r) is a member of edge group s_i, $L_r(K_r(c)) = i$. Since most pixels do not belong to an edge group, using these two data structures we can efficiently find the list of overlapping edge groups by searching L_r from index $K_r(c_0)$ to $K_r(c_1)$.

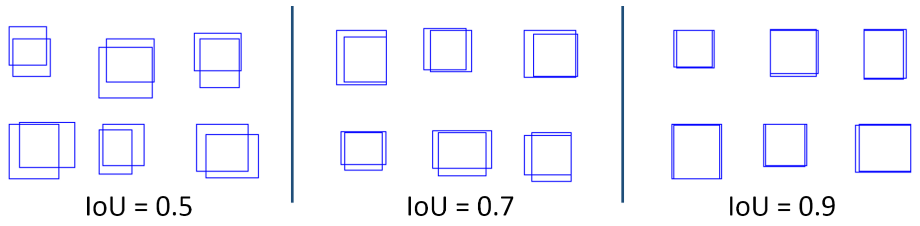

Fig. 2. An illustration of random bounding boxes with Intersection over Union (IoU) of 0.5, 0.7, and 0.9. An IoU of 0.7 provides a reasonable compromise between very loose (IoU of 0.5) and very strict (IoU of 0.9) overlap values.

3.4 Search strategy

When searching for object proposals, the object detection algorithm should be taken into consideration. Some detection algorithms may require object proposals with high accuracy, while others are more tolerant of errors in bounding box placement. The accuracy of a bounding box is typically measured using the Intersection over Union (IoU) metric. IoU computes the intersection of a candidate box and the ground truth box divided by the area of their union. When evaluating object detection algorithms, an IoU threshold of 0.5 is typically used to determine whether a detection was correct [15]. However as shown in Figure 2, an IoU score of 0.5 is quite loose. Even if an object proposal is generated with an IoU of 0.5 with the ground truth, the detection algorithm may provide a low score. As a result, IoU scores of greater than 0.5 are generally desired.

In this section we describe an object proposal search strategy based on the desired IoU, δ, for the detector. For high values of δ we generate a more concentrated set of bounding boxes with higher density near areas that are likely to contain an object. For lower values of δ the boxes can have higher diversity, since it is assumed the object detector can account for moderate errors in box location. Thus, we provide a tradeoff between finding a smaller number of objects with higher accuracy and a higher number of objects with less accuracy. Note that previous methods have an implicit bias for which δ they are designed for, e.g. Objectness [4] and Randomized Prim [8] are tuned for low and high δ, respectively, whereas we provide explicit control over diversity versus accuracy.

We begin our search for candidate bounding boxes using a sliding window search over position, scale and aspect ratio. The step size for each is determined using a single parameter α indicating the IoU for neighboring boxes. That is, the step sizes in translation, scale and aspect ratio are determined such that one step results in neighboring boxes having an IoU of α. The scale values range from a minimum box area of $\sigma = 1000$ pixels to the full image. The aspect ratio varies from $1/\tau$ to τ, where $\tau = 3$ is used in practice. As we discuss in Section 4, a value of $\alpha = 0.65$ is ideal for most values of δ. However, if a highly accurate $\delta > 0.9$ is required, α may be increased to 0.85.

After a sliding window search is performed, all bounding box locations with a score h_b^{in} above a small threshold are refined. Refinement is performed using

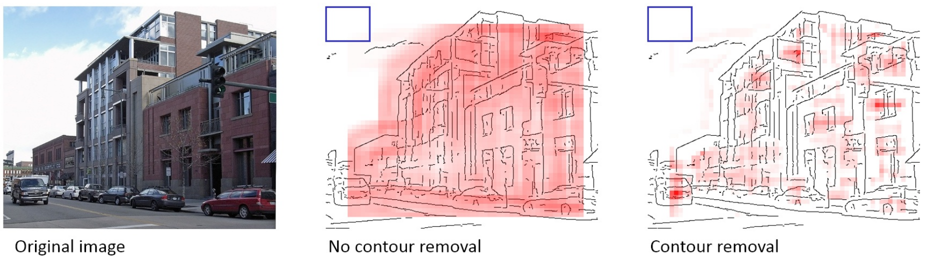

Fig. 3. Illustration of the computed score using (middle) and removing (right) contours that overlap the bounding box boundary. Notice the lack of clear peaks when the contours are not removed. The magnitudes of the scores are normalized for viewing. The box dimensions used for generating the heatmaps are shown by the blue rectangles.

a greedy iterative search to maximize h_b^{in} over position, scale and aspect ratio. After each iteration, the search step is reduced in half. The search is halted once the translational step size is less than 2 pixels.

Once the candidate bounding boxes are refined, their maximum scores are recorded and sorted. Our final stage performs Non-Maximal Suppression (NMS) of the sorted boxes. A box is removed if its IoU is more than β for a box with greater score. We have found that in practice setting $\beta = \delta + 0.05$ achieves high accuracy across all values of δ, Section 4.

4 Results

In this section we explore the performance and accuracy of our Edge Boxes algorithm in comparison to other approaches. Following the experimental setup of previous approaches [7, 5, 8, 4] we evaluate our algorithm on the PASCAL VOC 2007 dataset [15]. The dataset contains 9,963 images. All results on variants of our approach are reported on the validation set and our results compared to other approaches are reported on the test set.

4.1 Approach variants

We begin by testing various variants of our approach on the validation set. Figure 4(a, b) illustrates the algorithm's behavior based on the parameters α and β that control the step size of the sliding window search and the NMS threshold, respectively, when generating 1000 object proposals.

As α is increased, the density of the sampling is increased, resulting in more candidate boxes being evaluated and slower runtimes, Table 1. Notice that the results for $\alpha = 0.65$ are better than or nearly identical to $\alpha = 0.70$ and $\alpha = 0.75$. Thus, if a lower IoU value δ is desired $\alpha = 0.65$ provides a nice accuracy vs. efficiency tradeoff. Depending on the desired IoU value of δ, the value of β may be adjusted accordingly. A value of $\beta = \delta + 0.05$ achieves high accuracy across all desired δ. As shown in Table 1, changes in β have minimal effect on runtime.

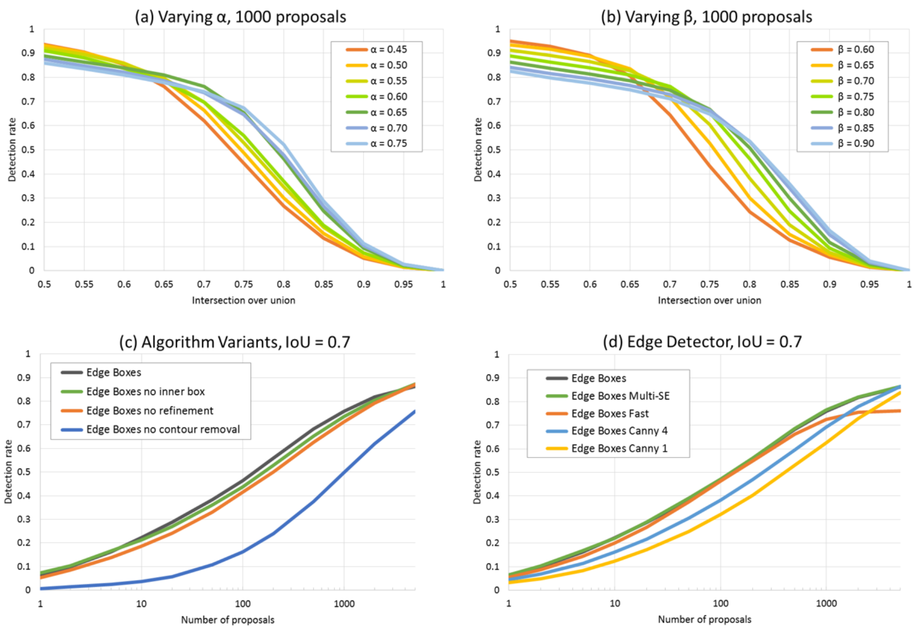

Fig. 4. A comparison of various variants of our approach. (a) The detection rate when varying the parameter α that varies the density of the sampling rate (default $\alpha = 0.65$). (b) Results while varying the parameter β controlling the NMS threshold (default $\beta = 0.75$). (c) The detection accuracy when various stages are removed from the algorithm, including the removal of edges in the inner box, the bounding box location refinement, and the removal of the contours that overlap the box's boundaries. (d) Detection accuracy when different edge detectors are used, including single-scale Structured Edges [16] (default), multi-scale Structure Edges, and a fast variant that runs at 10 fps without the edge sharpening enhancement introduced in [19]. Results using the Canny edge detector [28] with varying amounts of blur are also shown.

Three useful variants of our algorithm are shown in Table 1; Edge Boxes 50, Edge Boxes 70, and Edge Boxes 90 that have settings for α and β adjusted for IoU thresholds of $\delta = 0.5, 0.7$ and 0.9 respectively. For higher IoU thresholds that require extremely tight bounding boxes, α must be adjusted to search more densely resulting in longer runtimes. Otherwise, α may be kept fixed.

Our second set of experiments tests several variants of the algorithm, Figure 4(c, d). For these experiments we set δ to an intermediate value of 0.7 and show detection rates when varying the number of object proposals. The primary contribution of our paper is that contours that overlap the bounding box's boundary should be removed when computing the box's score. Figure 4 shows that if these contours are not removed a significant drop in accuracy is observed. To gain intuition into the effect of the contour removal on the score, we illustrate the score computed with and without contour removal in Figure 3. With contour

	IoU = 0.5		IoU = 0.7		IoU = 0.9				
	AUC	Recall	AUC	Recall	AUC	Recall	Runtime	α	β
Edge boxes 50	**.64**	**96%**	.36	55%	.04	5%	**.25s**	.65	.55
Edge boxes 70	.58	89%	**.45**	**76%**	.06	9%	**.25s**	.65	.75
Edge boxes 90	.38	59%	.28	46%	**.15**	**28%**	2.5s	.85	.95

Table 1. Accuracy measures and runtimes for three variants of our algorithm: Edge Boxes 50, Edge Boxes 70 and Edge Boxes 90. Accuracy measures include Area Under the Curve (AUC) and proposal recall at 1000 proposals. Parameter values for α and β are shown. All other parameters are held constant.

removal the scores have strong peaks around clusters of edges that are more likely to form objects given the current box's size. Notice the strong peak with contour removal in the bottom left-hand corner corresponding to the van. When contours are not removed strong responses are observed everywhere.

If the center edges are not removed, using h_b instead of h_b^{in}, a small drop in accuracy is found. Not performing box location refinement results in a more significant drop in accuracy. The quality of the initial edge detector is also important, Figure 4(d). If the initial edge map is generated using gradient-based Canny edges [28] with varying blur instead of Structured Edges [16] the results degrade. If multi-scale Structured Edges are computed instead of single-scale edges there is a minimal gain in accuracy. Since single-scale edges can be computed more efficiently we use single-scale edges for all remaining experiments. The runtime of our baseline approach which utilizes single-scale edges is 0.25s.

If near real-time performance is desired, the parameters of the algorithm may be adjusted to return up to 1000 boxes with only a minor loss in accuracy. Specifically, we can reduce α to 0.625, and increase the threshold used to determine which boxes to refine from 0.01 to 0.02. Finally, if we also disable the sharpening enhancement of the Structured Edge detector [19], the runtime of our algorithm is 0.09s. As shown in Figure 4(d), this variant, called Edge Boxes Fast, has nearly identical results when returning fewer than 1000 boxes.

4.2 Comparison with state-of-the-art

We compare our Edge Boxes algorithm against numerous state-of-the-art algorithms summarized in Table 2. Results of all competing methods were provided by Hosang et al. [26] in a standardized format. Figure 5 (top) shows the detection rates when varying the number of object proposals for different IoU thresholds. For each plot, we update our parameters based on the desired value of δ using the parameters in Table 1. Edge Boxes performs well across all IoU values and for both a small and large number of candidates. Selective Search [5] achieves competitive accuracy, especially at higher IoU values and larger number of boxes. CPMC [6] generates high quality proposals but produces relatively few candidates and is thus unable to achieve high recall. BING [11], which is very fast, generates only very loosely fitting proposals and hence is only competi-

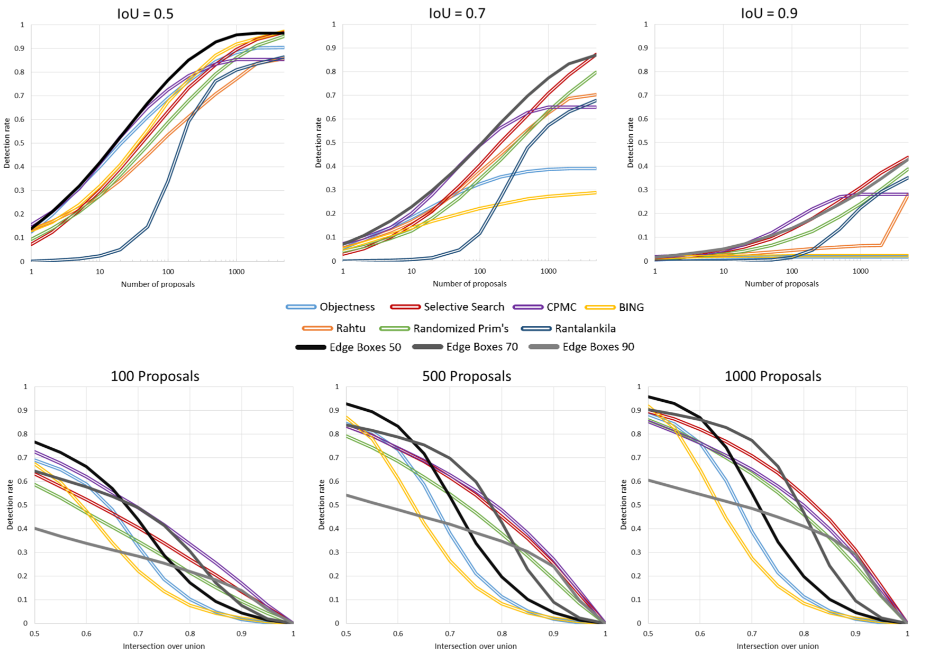

Fig. 5. Comparison of Edge Boxes to various state-of-the-algorithms, including Objectness [4], Selective Search [5], Randomized Prim's [8] and Rahtu [7]. The variations of our algorithm are tested using $\delta = 0.5, 0.7$ and 0.9 indicated by Edge Boxes 50, Edge Boxes 70 and Edge Boxes 90. (top) The detection rate vs. the number of bounding box proposals for various intersection over union thresholds. (bottom) The detection rate vs. intersection over union for various numbers of object proposals.

tive at low IoU. In contrast our approach achieves good results across a variety of IoU thresholds and quantity of object proposals. In fact, as shown in Table 2, to achieve a recall of 75% with an IoU of 0.7 requires 800 proposals using Edge Boxes, 1400 proposals using Selective Search, and 3000 using Randomized Prim's. No other methods achieve 75% recall using even 5000 proposals. Edge Boxes also achieves a significantly higher maximum recall (87%) and Area Under the Curve (AUC = 0.46) as compared to all approaches except Selective Search.

Figure 5 (bottom) shows the detection rate when varying the IoU threshold for different numbers of proposals. Similar to Figure 4(a,b), these plots demonstrate that setting parameters based on δ, the desired IoU threshold, leads to good performance. No single algorithm or set of parameters is capable of achieving superior performance across all IoU thresholds. However, Edge Boxes 70 performs well over a wide range of IoU thresholds that are typically desired in practice (IoU between 0.5 and 0.8, Figure 2). Segmentation based methods along with Edge Boxes 90 perform best at very high IoU values.

We compare the runtime and summary statistics of our approach to other methods in Table 2. The runtimes for Edge Boxes includes the 0.1 seconds needed

	AUC	N@25%	N@50%	N@75%	Recall	Time
BING [11]	.20	292	–	–	29%	**.2s**
Rantalankila [10]	.23	184	584	–	68%	10s
Objectness [4]	.27	27	–	–	39%	3s
Rand. Prim's [8]	.35	42	349	3023	80%	1s
Rahtu [7]	.37	29	307	–	70%	3s
Selective Search [5]	.40	28	199	1434	**87%**	10s
CPMC [6]	.41	15	111	–	65%	250s
Edge boxes 70	**.46**	**12**	**108**	**800**	**87%**	**.25s**

Table 2. Results for our approach, Edge Boxes 70, compared to other methods for IoU threshold of 0.7. Methods are sorted by increasing Area Under the Curve (AUC). Additional metrics include the number of proposals needed to achieve 25%, 50% and 75% recall and the maximum recall using 5000 boxes. Edge Boxes is best or near best under every metric. All method runtimes were obtained from [26].

to compute the initial edges. Table 2 shows that our approach is significantly faster and more accurate than previous approaches. The only methods with comparable accuracy are Selective Search and CPMC, but these are considerably slower. The only method with comparable speed is BING, but BING has the worst accuracy of all evaluated methods at IoU of 0.7.

Finally, qualitative results are shown in Figure 6. Many of the errors occur with small or heavily occluded objects in the background of the images.

5 Discussion

In this paper we propose an effective method for finding object proposals in images that relies on one simple observation: the number of edges that are wholly enclosed by a bounding box is indicative of the likelihood of the box containing an object. We describe a straightforward scoring function that computes the weighted sum of the edge strengths within a box minus those that are part of a contour that straddles the box's boundary. Using efficient data structures and smart search strategies we can find object proposals rapidly. Results show both improved accuracy and increased efficiency over the state of the art.

One interesting direction for future work is using the edges to help generate segmentation proposals in addition to the bounding box proposals for objects. Many edges are removed when scoring a candidate bounding box; the location of these suppressed edges could provide useful information in generating segmentations. Finally we will work with Hosang at al. to add Edge Boxes to their recent survey and evaluation of object proposal methods [26] and we also hope to evaluate our proposals coupled with state-of-the-art object detectors [13].

Source code for Edge Boxes will be made available online.

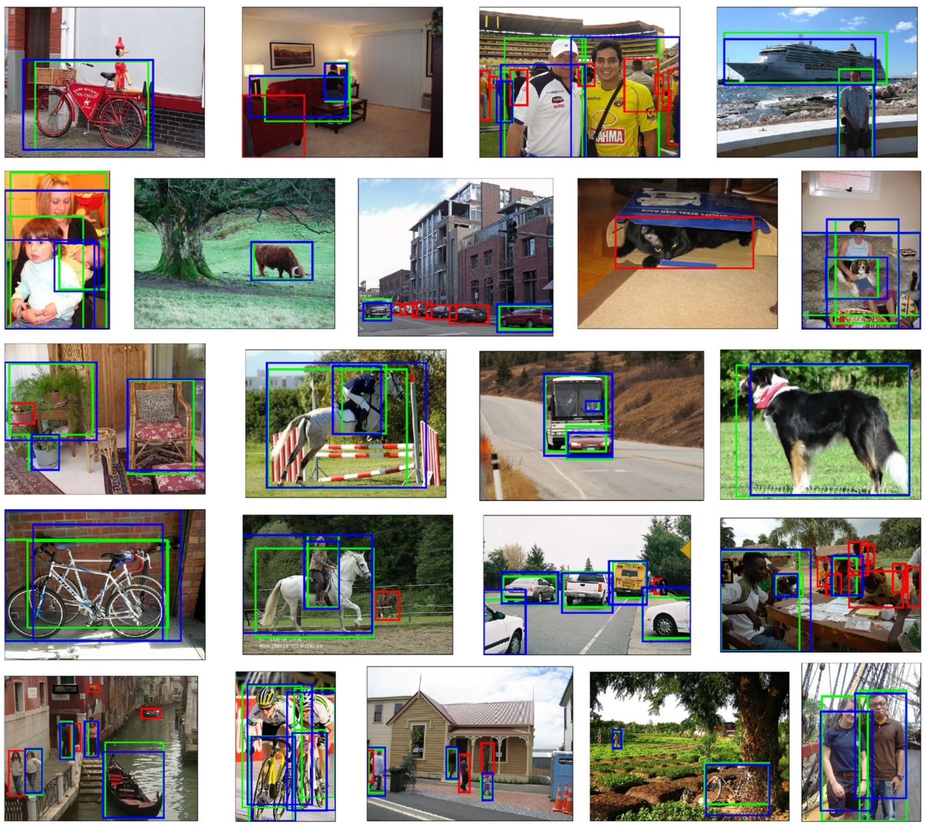

Fig. 6. Qualitative examples of our object proposals. Blue bounding boxes are the closest produced object proposals to each ground truth bounding box. Ground truth bounding boxes are shown in green and red, with green indicating an object was found and red indicating the object was not found. An IoU threshold of 0.7 was used to determine correctness for all examples. Results are shown for Edge Boxes 70 with 1,000 object proposals. At this setting our approach returns over 75% of object locations.

References

1. Viola, P.A., Jones, M.J.: Robust real-time face detection. IJCV **57**(2) (2004) 137–154
2. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
3. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI **32**(9) (2010) 1627–1645
4. Alexe, B., Deselaers, T., Ferrari, V.: Measuring the objectness of image windows. PAMI **34**(11) (2012)
5. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. IJCV (2013)

6. Carreira, J., Sminchisescu, C.: Cpmc: Automatic object segmentation using constrained parametric min-cuts. PAMI **34**(7) (2012)
7. Rahtu, E., Kannala, J., Blaschko, M.: Learning a category independent object detection cascade. In: ICCV. (2011)
8. Manen, S., Guillaumin, M., Van Gool, L., Leuven, K.: Prime object proposals with randomized prims algorithm. In: ICCV. (2013)
9. Endres, I., Hoiem, D.: Category-independent object proposals with diverse ranking. PAMI (2014)
10. Rantalankila, P., Kannala, J., Rahtu, E.: Generating object segmentation proposals using global and local search. In: CVPR. (2014)
11. Cheng, M.M., Zhang, Z., Lin, W.Y., Torr, P.: BING: Binarized normed gradients for objectness estimation at 300fps. In: CVPR. (2014)
12. Wang, X., Yang, M., Zhu, S., Lin, Y.: Regionlets for generic object detection. In: ICCV. (2013)
13. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)
14. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. (2009)
15. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV **88**(2) (2010) 303–338
16. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV. (2013)
17. Marr, D.: Vision: A computational investigation into the human representation and processing of visual information. Inc., New York, NY (1982)
18. Eitz, M., Hays, J., Alexa, M.: How do humans sketch objects? ACM Transactions Graphics **31**(4) (2012)
19. Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. CoRR **abs/1406.5549** (2014)
20. Deselaers, T., Alexe, B., Ferrari, V.: Localizing objects while learning their appearance. In: ECCV. (2010)
21. Siva, P., Xiang, T.: Weakly supervised object detector learning with model drift detection. In: ICCV. (2011)
22. Gu, C., Lim, J.J., Arbeláez, P., Malik, J.: Recognition using regions. In: CVPR. (2009)
23. Hoiem, D., Efros, A.A., Hebert, M.: Geometric context from a single image. In: ICCV. (2005)
24. Russell, B.C., Freeman, W.T., Efros, A.A., Sivic, J., Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: CVPR. (2006)
25. Malisiewicz, T., Efros, A.A.: Improving spatial support for objects via multiple segmentations. In: BMVC. (2007)
26. Hosang, J., Benenson, R., Schiele, B.: How good are detection proposals, really? In: BMVC. (2014)
27. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. IJCV **59**(2) (2004)
28. Canny, J.: A computational approach to edge detection. PAMI (6) (1986) 679–698