# Measuring the objectness of image windows

Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari

**Abstract**—We present a generic objectness measure, quantifying how likely it is for an image window to contain an object of any class. We explicitly train it to distinguish objects with a well-defined boundary in space, such as cows and telephones, from amorphous background elements, such as grass and road. The measure combines in a Bayesian framework several image cues measuring characteristics of objects, such as appearing different from their surroundings and having a closed boundary. These include an innovative cue to measure the closed boundary characteristic. In experiments on the challenging PASCAL VOC 07 dataset, we show this new cue to outperform a state-of-the-art saliency measure, and the combined objectness measure to perform better than any cue alone. We also compare to interest point operators, a HOG detector, and three recent works aiming at automatic object segmentation. Finally, we present two applications of objectness. In the first, we sample a small number windows according to their objectness probability and give an algorithm to employ them as location priors for modern class-specific object detectors. As we show experimentally, this greatly reduces the number of windows evaluated by the expensive class-specific model. In the second application, we use objectness as a complementary score in addition to the class-specific model, which leads to fewer false positives. As shown in several recent papers, objectness can act as a valuable focus of attention mechanism in many other applications operating on image windows, including weakly supervised learning of object categories, unsupervised pixelwise segmentation, and object tracking in video. Computing objectness is very efficient and takes only about 4 sec. per image.

**Index Terms**—Objectness Measure, Object Detection, Object Recognition.

## 1 Introduction

In recent years object class detection has become a major research area. Although a variety of approaches exist [4, 11, 35], most state-of-the-art detectors follow the sliding-window paradigm [11, 12, 18, 25, 33]. A classifier is first trained to distinguish windows containing instances of a given class from all other windows. The classifier is then used to score every window in a test image. Local maxima of the score localize instances of the class.

While object detectors are specialized for one object class, such as cars or swans, in this paper we define and train a measure of objectness *generic over classes*. It quantifies how likely it is for an image window to cover an object of *any* class. Objects are standalone things with a well-defined boundary and center, such as cows, cars, and telephones, as opposed to amorphous background stuff, such as sky, grass, and road (as in the things versus stuff distinction of [26]). Fig. 1 illustrates the desired behavior of an objectness measure. It should score highest windows fitting an object tight (green), score lower windows covering partly an object and partly the background (blue), and score lowest windows containing only stuff (red).

In order to define the objectness measure, we argue that any object has at least one of three distinctive characteristics:

- a well-defined *closed boundary* in space
- a *different appearance* from its surroundings [36, 38]
- sometimes it is *unique* within the image and stands out as salient [7, 21, 27, 28]

- *B. Alexe is with the Computer Vision Laboratory at ETH Zurich. E-mail: bogdan@vision.ee.ethz.ch*
- *T. Deselaers is with Google Zurich. E-mail: deselaers@gmail.com*
- *V. Ferrari is with the Computer Vision Laboratory at ETH Zurich. E-mail: ferrari@vision.ee.ethz.ch*

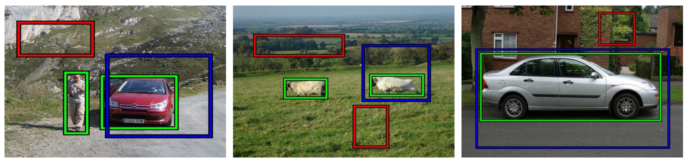

(a) (b) (c)

Fig. 1: **Desired behavior of an objectness measure.** *The objectness measure should score the blue windows, partially covering the objects, lower than the ground truth windows (green), and score even lower the red windows containing only stuff.*

Many objects have several of these characteristics at the same time (fig. 2-6). In sec. 2.1 to 2.4 we explore several image cues designed to measure these characteristics.

Although objects can appear at a variety of locations and sizes within an image, some windows are more likely to cover objects than others, even without analyzing the pixel patterns inside them. A very elongated window in an image corner is less probable *a priori* than a square window in the middle of it. We explore this idea in sec. 2.5 and propose an objectness cue based on the location and size of a window.

This paper makes several contributions: (a) We design an objectness measure and explicitly train it to distinguish windows containing an object from background windows. This measure combines in a Bayesian framework several cues based on the above characteristics (secs. 2 and 3). (b) On the task of detecting objects in the challenging PASCAL VOC 07 dataset [17], we demonstrate that the combined objectness measure performs better than any cue alone, and also outperforms traditional salient blob detectors [27, 28], interest point detectors [41], the Semantic Labeling technique of [22], and a HOG detector [11] trained to detect objects of arbitrary classes. (c) We give an algorithm that employs objectness to greatly reduce the number of evaluated windows evaluated by modern class-specific object detectors [11, 18, 50] (sec. 6). Different from ESS [33], our method imposes no restriction on the class model used to score a window. (d) Analoguosly, we show how to use objectness for reducing the number of false-positives returned by class-specific object detectors (sec. 7).

In addition to the two applications above, objectness is valuable in others as well. Since the publication of a preliminary version of this work [3], objectness has already been used successfully to (a) facilitate learning new classes in a weakly supervised scenario [13, 30, 47], where the location of object instances is unknown. Objectness steers the localization algorithm towards objects rather than backgrounds, e.g. avoiding to select only grass regions in a set of sheep images; (b) analoguously, to support weakly supervised pixelwise segmentation of object classes [2, 52] and unsupervised object discovery [34]; (c) learning spatial models of interactions between humans and objects [45]; (d) content-aware image resizing [5, 48]; (e) object tracking in video, as an additional likelihood term preventing the tracker from drifting to the background (ongoing work by Van Gool's group at ETH Zurich). The source code of the objectness measure is available at http://www.vision.ee.ethz.ch/~calvin. Computing objectness is very efficient and takes only about 4 sec. per image

## 1.1 Related work

This paper is related to several research strands, which differ in how they define 'saliency'.

**Interest points.** Interest point detectors (IPs) [29, 41] respond to local textured image neighborhoods and are widely used for finding image correspondences [41] and recognizing specific objects [37]. IPs focus on individual points, while our approach is trained to respond to entire objects. Moreover, IPs are designed for repeatable detection in spite of changing imaging conditions, while our objectness measure is trained to distinguish objects from backgrounds. In sec. 5, we experimentally evaluate IPs on our task.

**Class-specific saliency.** A few works [40, 42, 53] define as salient the visual characteristics that best distinguish a particular object class (e.g. cars) from others. This class-specific saliency is very different from the class-generic task we tackle here.

**Generic saliency.** Since [28], numerous works [7, 21, 24, 27, 31] appeared to measure the saliency of pixels, as the degree of uniqueness of their neighborhood wrt the entire image or the surrounding area [36, 38]. Salient pixels form blobs that 'stand out' from the image. These works implement selective visual attention from a bottom-up perspective and are often inspired by studies of human eye movements [14, 15, 32].

Liu et al. [36] detect a single dominant salient object in an image. They formulate the problem as image segmentation. The method combines local, regional and global pixel-based saliency measurements in a CRF and derives a binary segmentation separating the object from the background. Valenti et al. [49] measure the saliency of individual pixels, and then returns the region with the highest sum of pixel-saliency as the single dominant object. Achanta et al. [1] find salient regions using a frequency-tuned approach. Their approach segments objects by adaptive thresholding of fully resolution saliency maps. The saliency maps are obtained by combining several band-pass filters that retain a wider range of spatial frequencies than [24, 27, 38]. Berg and Berg [6] find iconic images representative for an object class. Their approach returns images having a large, centered object, which is clearly separated from the background. These approaches do not seem suitable for the PASCAL VOC 07 dataset [17] where many objects are present in an image and they are rarely dominant (fig. 15, 16).

This paper is related to the above works, as we are looking for generic objects. We incorporate a state-of-the-art saliency detector [27] as one cue into our objectness measure. However, we also include other cues than 'stand out' saliency and demonstrate that our combined measure performs better at finding objects (sec. 5).

Our work differs from the above also in other respects. The unit of analysis is not a *pixel*, as possibly belonging to an object, but a *window*, as possibly containing an entire object. This enables scoring all windows in an image and sampling any desired number of windows according to their scores. These can then directly be fed as useful location priors to object class learning and detection algorithms, rather than making hard decisions early on. We experimentally demonstrate this with applications to speeduping object detectors (sec. 6) and to reducing their false-positive rates (sec. 7).

Analyzing windows also enables evaluating more complex measures of objectness. For example, in sec. 2.4 we propose a new image cue and demonstrate it performs better than traditional saliency cues [27] at finding entire objects.

Finally, we evaluate our method on a more varied and challenging dataset (PASCAL VOC 07), where most images contain many objects and they appear over a wide range of scales. We explicitly train our objectness measure to satisfy the strict PASCAL-overlap criterion, and evaluate its performance using it. This matters because it is the standard criterion for evaluating the intended clients of our objectness measure, i.e. object detection algorithms.

**Object proposals.** Since the publication of the first version of this work [3], two closely related independent works appeared [9, 16]. They share with our work the overall idea of proposing a few hundred class-independent candidates likely to cover all objects in an image. While both works [9, 16] produce rough segmentations as object candidates, they are computationally very expensive and take 2-7 minutes per image (compared to a few seconds for our method). In sec. 5 we compare the ability of our method to detect generic objects to [9, 16].

**Efficient sliding window object detection.** State-of-the-art object detection algorithms often rely on the sliding window paradigm [11, 18, 50], which scores a large number of windows in a test image using a classifier. To keep the computational cost manageable, researchers are careful in selecting classifiers that can be evaluated rapidly (e.g. linear SVMs [11, 20]).

Recently, several techniques have been proposed to reduce the cost of sliding window algorithms. One approach is to

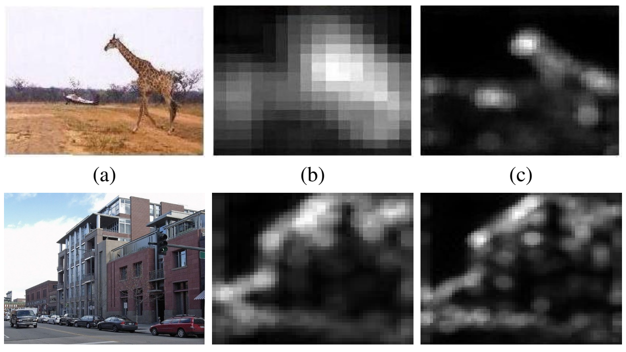

(a) (b) (c) (d) (e) (f)

Fig. 2: **MS success and failure. Success:** *the large giraffe in the original image (a) appears as a blob in the saliency map for a high scale (b), while the tiny airplane in the map for a low scale (c). Having multi-scale saliency maps is important for finding more objects in challenging datasets. Interestingly, at the low scale the head of the giraffe is salient, rather than the whole giraffe.* **Failure:** *the numerous cars in the original image (d) are not salient at any scale. We show the saliency maps for 2 scales in (e) and (f). The contour of the building appears more salient than the cars.*

reduce the cost of evaluating a complex classifier on a window. Excellent approximations to the histogram intersection kernel [39] and the $\chi^2$ kernel [51] have been presented. Another approach is to reduce the number of times the complex classifier is evaluated. In [25, 50] this is achieved by first running a linear classifier over all windows, and then evaluating a non-linear kernel classifier only on a few highly scored windows. Lampert et al. [33] presented a branch-and-bound scheme that exactly finds the single highest scored window while minimizing the number of times the classifier is evaluated.

In sec. 6 we apply objectness to greatly reduce the number of windows evaluated by the classifier. Our method can be applied to *any* window classifier, whereas [33] is restricted to those for which the user can provide a good bound on the highest score in a contiguous set of windows. Our speedup is orthogonal to techniques to reduce the cost of evaluating one window, and could be combined with them [25, 39, 50, 51].

### 1.2 Plan of the paper

Sec. 2 describes the cues composing our objectness measure. Secs. 3 and 4 show how to learn the cue parameters and how to combine them in a Bayesian framework. In sec. 5 we evaluate objectness on the very challenging PASCAL VOC 2007 dataset, and then show applications to aiding class-specific object detectors [11, 18, 50] in secs. 6 and 7.

## 2 Objectness cues

As mentioned in sec. 1, objects in an image are characterized by a closed boundary in 3D space, a different appearance from their immediate surrounding and sometimes by uniqueness. In secs. 2.1 to 2.4 we present four image cues to measure these characteristics for an image window. As a complementary strategy, we define a last cue based on the location and size of a window, without analyzing the actual image pixels (sec. 2.5).

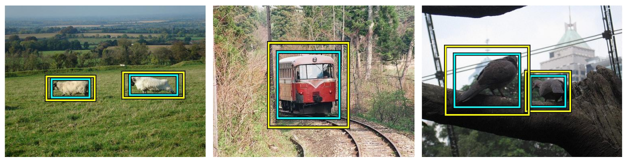

(a) (b) (c)

Fig. 3: **CC success and failure. Success:** *the windows containing the objects (cyan) have high color contrast with their surrounding ring (yellow) in images (a) and (b).* **Failure:** *the color contrast for the object in image (c) is much lower. The surrounding ring drawn has the optimal size $\theta^*_{CC}$, as learned in sec. 3.*

### 2.1 Multi-scale Saliency (MS)

Hou et al. [27] proposed a global saliency measure based on the spectral residual of the FFT, which favors regions with an unique appearance within the entire image $f$. The saliency map $I$ of an image $f$ is obtained at each pixel $p$ as

$$I(p) = g(p) * \mathcal{F}^{-1}\Big[\exp\big(\mathcal{R}(f) + \mathcal{P}(f)\big)\Big]^2 \tag{1}$$

where $\mathcal{F}$ is the FFT, $\mathcal{R}(f)$ and $\mathcal{P}(f)$ are the spectral residual and the phase spectrum [27] of the image $f$, and $g$ is a Gaussian filter used for smoothing the output. Since this measure prefers objects at a certain scale, we extend it to multiple scales (fig. 2). Moreover, as [27] suggests, we process the color channels independently as separate images. For each scale $s$, we use [27] to obtain a saliency map $I^s_{\mathrm{MS}}(p)$. Based on this, we define the saliency of a window $w$ at scale $s$ as

$$\mathrm{MS}(w, \theta^s_{\mathrm{MS}}) = \sum_{\{p \in w | I^s_{\mathrm{MS}}(p) \geq \theta^s_{\mathrm{MS}}\}} I^s_{\mathrm{MS}}(p) \times \frac{|\{p \in w | I^s_{\mathrm{MS}}(p) \geq \theta^s_{\mathrm{MS}}\}|}{|w|} \tag{2}$$

where the scale-specific thresholds $\theta^s_{\mathrm{MS}}$ are parameters to be learned (sec. 3), and $|\cdot|$ indicates the number of pixels. Saliency is higher for windows with higher density of salient pixels (second factor), with a bias towards larger windows (first factor). Density alone would score highest windows comprising just a few very salient pixels. Instead, our measure is designed to score highest windows around entire *blobs* of salient pixels, which correspond better to whole objects (fig. 2). The need for multiple scales is evident in fig. 2 as the windows covering the two objects in the image (airplane, giraffe) score highest at different scales. This MS cue measures the *uniqueness* characteristic of objects.

### 2.2 Color Contrast (CC)

CC is a measure of the dissimilarity of a window to its immediate surrounding area (fig. 3). The surrounding $\mathrm{Surr}(w, \theta_{CC})$ of a window $w$ is a rectangular ring obtained by enlarging the window by a factor $\theta_{CC}$ in all directions, so that

$$\frac{|\mathrm{Surr}(w, \theta_{CC})|}{|w|} = \theta^2_{CC} - 1 \tag{3}$$

The CC between a window and its surrounding is computed as the Chi-square distance between their LAB histograms $h$

$$\mathrm{CC}\left(w, \theta_{CC}\right) = \chi^2(h(w),\ h(\mathrm{Surr}(w, \theta_{CC}))) \tag{4}$$

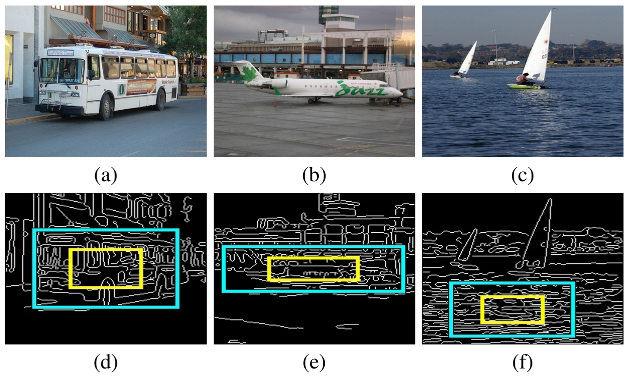


Fig. 4: **ED success and failure. Success:** *given images (a) and (b) the cyan windows covering the bus and the aeroplane score high as the density of edges is concentrated in these regions.* **Failure:** *in image (c) the cyan window along with many other windows covering the water score high determining a high rate of false positives. In particular the windows covering the boats have a low score. We show the Canny edge maps in (d), (e) and (f) for a value $\theta_{ED} = 2$ (the optimal value $\theta^*_{ED}$ goes to infinity, shrinking the inner ring to a point).*

The ring size parameter $\theta_{CC}$ is learned in sec. 3.

CC is a useful cue because objects tend to have a different appearance (color distribution) than the background behind them. In fig. 3a, windows on the grass score lower than windows half on a sheep and half on the grass. Windows fitting a sheep tightly score highest. This cue measures the *different appearance* characteristic of objects.

CC is related to the center-surround histogram cue of [36]. However, [36] computes a center-surround histogram centered at a pixel, whereas CC scores a whole window as whether it contains an entire object. The latter seems a more appropriate level of analysis.

### 2.3 Edge Density (ED)

ED measures the density of edges near the window borders. The inner ring $\text{Inn}(w, \theta_{ED})$ of a window $w$ is obtained by shrinking it by a factor $\theta_{ED}$ in all directions, so that

$$\frac{|\text{Inn}(w, \theta_{ED})|}{|w|} = \frac{1}{\theta^2_{ED}} \tag{5}$$

The ED of a window $w$ is computed as the density of edgels[1] in the inner ring

$$\text{ED}(w, \theta_{ED}) = \frac{\sum_{p \in \text{Inn}(w,\theta_{ED})} I_{ED}(p)}{\text{Len}(\text{Inn}(w, \theta_{ED}))} \tag{6}$$

The binary edgemap $I_{ED}(p) \in \{0, 1\}$ is obtained using the Canny detector [8], and $\text{Len}(\cdot)$ measures the perimeter of the inner ring. Note how the expected number of boundary edgels grows proportionally to the perimeter, not the area, because edgels have constant thickness of 1 pixel. This normalization avoids a bias towards very small windows that would otherwise arise. The ring size parameter $\theta_{ED}$ is learned in sec. 3.

The ED cue captures the *closed boundary* characteristic of objects, as they tend to have many edgels in the inner ring (fig. 4).

[1]an edgel is a pixel classified as edge by an edge detector

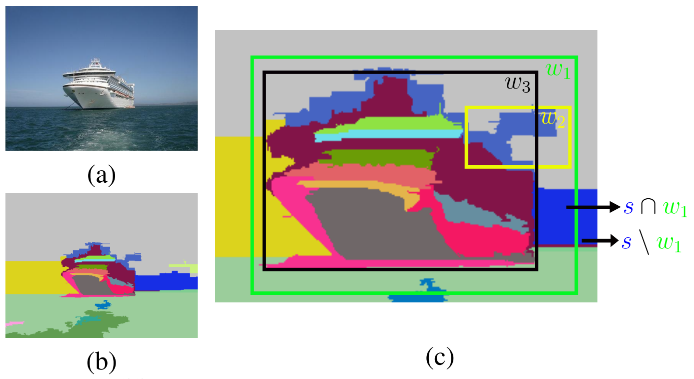


Fig. 5: **The SS cue.** *Given the segmentation (b) of image (a), for a window $w$ we compute $\text{SS}(\text{w}, \theta_{\text{SS}})$ (eq. 7). In (c), most of the surface of $w_1$ is covered by superpixels contained almost entirely inside it. Instead, all superpixels passing by $w_2$ continue largely outside it. Therefore, $w_1$ has a higher SS score than $w_2$. The window $w_3$ has an even higher score as it fits the object tightly.*

### 2.4 Superpixels Straddling (SS)

A different way of capturing the *closed boundary* characteristic of objects rests on using superpixels [19] as features. Superpixels segment an image into small regions of uniform color or texture. A key property of superpixels is to preserve object boundaries: all pixels in a superpixel belong to the same object [46] (ideally). Hence, an object is typically oversegmented into several superpixels, but none straddles its boundaries (fig. 5). Based on this property, we propose here a cue to estimate whether a window covers an object.

A superpixel $s$ straddles a window $w$ if it contains at least one pixel inside and at least one outside $w$. Most of the surface of an object window is covered by superpixels contained entirely inside it ($w_1$ in fig. 5c). Instead, most of the surface of a 'bad' window is covered by superpixels straddling it (i.e. superpixels continuing outside the window, $w_2$ in fig. 5c). The SS cue measures for all superpixels $s$ the degree by which they straddle $w$

$$\text{SS}(w, \theta_{SS}) = 1 - \sum_{s \in \mathcal{S}(\theta_{SS})} \frac{\min(|s \setminus w|, |s \bigcap w|)}{|w|} \tag{7}$$

where $\mathcal{S}(\theta_{SS})$ is the set of superpixels obtained using [19] with a segmentation scale $\theta_{SS}$ (learned in sec. 3). For each superpixel $s$, eq. (7) computes its area $|s \bigcap w|$ inside $w$ and its area $|s \setminus w|$ outside $w$. The minimum of the two is the degree by which $s$ straddles $w$ and is its contribution to the sum in eq. (7).

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

In sec. 5 we show that SS outperforms all other cues we consider (including MS and its original form [27]).

**Efficient computation of SS.** Naively computing the score $\text{SS}(w, \theta_{SS})$ for window $w$ in an image $I$ requires $O(|I|)$ operations. For each superpixel in the whole image we need to sum over all the pixels it contains to compute eq. (7).

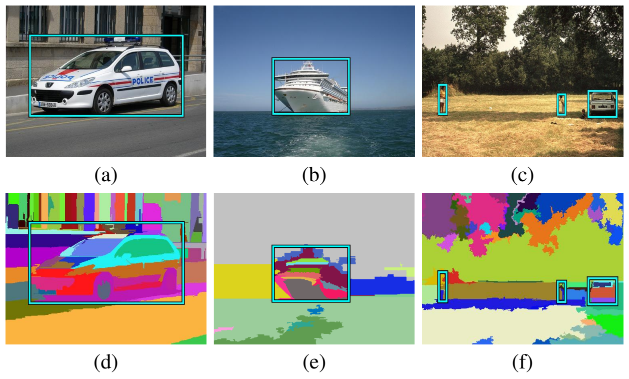


Fig. 6: **SS success and failure. Success**: *The cyan windows in (a) and (b) have high* SS *score computed from segmentations (d) and (e).* **Failure**: *Segmentation produces superpixels (f) not preserving the boundaries of the small objects in (c), resulting in low* SS. *All segmentations are obtained using the optimal segmentation scale $\theta^*_{SS}$ learned in sec. 3.*

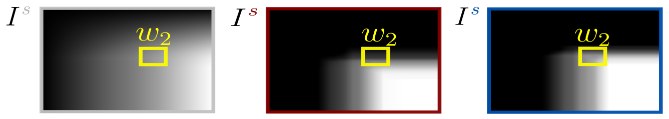


Fig. 7: **Efficiently computing eq. (7)**. *Window $w_2$ from fig. 5c is straddled by three superpixels. We show here the corresponding three integral images $\mathrm{I^s}$ as grayscale maps. The individual superpixel contributions $\frac{\min(|s\setminus w_2|,|s\bigcap w_2|)}{|w_2|}$ are computed based on them.*

Hence, the total cost of computing the score $\mathrm{SS}(w,\theta_{SS})$ for all windows $w$ in a set $W$ is $O(|W||I|)$ operations.

We present here an efficient procedure to reduce this complexity. For each superpixel $s$ we build a separate integral image $\mathrm{I^s(x,y)}$ giving the number of pixels of $s$ in the rectangle $(0,0)\rightarrow(x,y)$. Then, using $\mathrm{I^s(x,y)}$, we can compute the number $|s\bigcap w|$ of pixels of $s$ contained in any window $w$ in constant time, independent of its area (fig. 7). The area $|s\setminus w|$ outside $w$ is also readily obtained as

$$|s\setminus w| = |s| - |s\bigcap w| \tag{8}$$

Therefore, we can efficiently compute all elements of $\mathrm{SS}(w,\theta_{SS})$ using just 4 operations per superpixel, giving a a total of $4S$ operations for a window (with $S=|\mathcal{S}(\theta_{SS})|$ the number of superpixels in the image).

Computing all integral images $\mathrm{I^s}$ takes $O(S|I|)$ operations and it is done only once (not for each window). Therefore, our efficient procedure takes only $O(S(|W|+|I|))$ operations, compared to $O(|W||I|)$ for the naive algorithm. This is much faster because $S<<|I|$. In sec. 5 we use this procedure for computing eq. 7 for 100000 windows in an image.

### 2.5 Location and Size (LS)

Windows covering objects vary in size and location within an image. However, some windows are more likely to cover objects than others: an elongated window located at the top of the image is less probable *a priori* than a square window in the image center. Based on this intuition, we propose a new cue to asses how likely an image window is to cover an object, based only on its location and size, not on the pixels inside it.

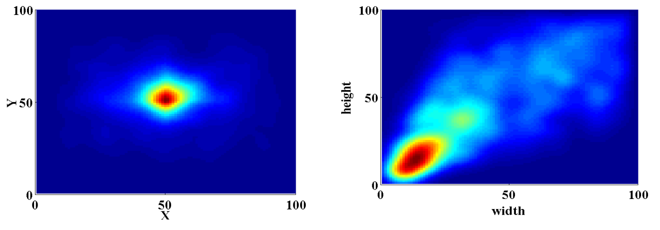


Fig. 8: **LS distributions.** *We visualize here two marginalizations of the* 4D *distribution over the training windows, each obtained by marginalizing over two dimensions. Left: over $(x,y)$ location. Right: over $(width, height)$. The location of object windows is Gaussian distributed, with mean at the centre of the image and a rather large spread. The distribution of object window sizes shows a positive correlation of width with height, suggesting that square windows have higher probability.*

We compute the probability of an image window to cover an object using kernel density estimation [44] in the 4D space $\mathcal{W}$ of all possible windows in an image. The space $\mathcal{W}$ is parametrized by the $(x,y)$ coordinates of the center, the width, and the height of a window. We equip $\mathcal{W}$ with a probability measure $p_{\mathcal{W}}$ defining how likely a window is to cover an object. This probability $p_{\mathcal{W}}$ is computed by kernel density estimation on a large training set of $N$ windows $\{w_1,w_2,\ldots,w_N\}$ covering objects, which are points in $\mathcal{W}$. As training images might have different sizes, we normalize the coordinate frame of all images to $100\times100$ pixels.

We perform kernel density estimation in the space $\mathcal{W}\subset[0,100]^4$ by placing a Gaussian with mean $w_i$ and covariance matrix $\theta_{LS}$ for each training window $w_i$. The covariance matrix $\theta_{LS}$ is learned as explained in sec. 3.3. The probability of a test window $w\in\mathcal{W}$ is computed as

$$p_{\mathcal{W}}(w,\theta_{LS}) = \frac{1}{Z}\sum_{i=1}^{N}\frac{1}{(2\pi)^2|\theta_{LS}|^{\frac{1}{2}}}e^{-\frac{1}{2}(w-w_i)^T(\theta_{LS})^{-1}(w-w_i)} \tag{9}$$

where the normalization constant $Z$ ensures that $p_{\mathcal{W}}$ is a probability, i.e. $\sum_{w\in\mathcal{W}}p_{\mathcal{W}}(w)=1$. Note that the query window $w$ can be *any* point in $\mathcal{W}$, not necessarily corresponding to a training window. The LS cue assesses the likelihood of a window $w$ to cover an object as $\mathrm{LS}(w,\theta_{LS}) = p_{\mathcal{W}}(w,\theta_{LS})$.

**Efficient computation of LS.** In practice, we consider a discretized version of the 4D space $\mathcal{W}$ that contains only windows with integer coordinates in a normalized image of $100\times100$ pixels. We precompute (9) for all these windows, obtaining a lookup table. For a test window $w$ we find its nearest neighbour $w^{nn}$ in the discretized space $\mathcal{W}$ and read out its value from the lookup table. This procedure is very efficient, as it takes very few operations for each test window, compared to $N$ operations in eq. (9).

### 2.6 Implementation details

**MS.** For every scale $s\in\{16,24,32,48,64\}$ and channel $c$ we rescale the image to $s\times s$ and then compute $\mathrm{MS}(w,\theta^s_{MS})$ using one integral image [10] (indicating the sum over the saliency of pixels in a rectangle).

**CC.** We convert the image to the quantized LAB space $4 \times 8 \times 8$ and then compute $\mathrm{CC}(w, \theta_{CC})$ using one integral image per quantized color.
**ED.** We rescale the image to $200 \times 200$ pixels and then compute $\mathrm{ED}(w, \theta_{ED})$ using one integral image (indicating the number of edgels in a rectangle).
**SS.** We obtain superpixels $\mathcal{S}(\theta_{SS})$ using the algorithm of [19] with segmentation scale $\theta_{SS}$. We efficiently compute $\mathrm{SS}(w, \theta_{SS})$ using one integral image per superpixel as detailed in sec. 2.4.

## 3 Learning cue parameters

We learn the parameters of the objectness cues from a training dataset $\mathcal{T}$ consisting of 1183 images from the PASCAL VOC 07 train+val dataset [17]. In PASCAL VOC 07 each image is annotated with ground-truth bounding-boxes for all objects from twenty categories (boat, bicycle, horse, etc). We use for training all images containing only objects of the 6 classes *bird, car, cat, cow, dog, sheep*, for a total of 1587 instances $\mathcal{O}$ (objects marked as difficult are ignored).

The parameters to be learned are $\theta_{CC}$, $\theta_{ED}$, $\theta_{SS}$, $\theta_{LS}$ and $\theta^s_{MS}$ (for 5 scales $s$). The first three are learned in a unified manner (sec. 3.1), whereas specialized methods are given for $\theta_{MS}$ (sec. 3.2) and $\theta_{LS}$ (sec. 3.3).

### 3.1 Learning the parameters of CC, ED, SS

We learn $\theta_{CC}$, $\theta_{ED}$, $\theta_{SS}$ in a Bayesian framework. As all three are learned in the same manner, we restrict the explanation to $\theta = \theta_{CC}$. For every image $I$ in $\mathcal{T}$ we generate 100000 random windows uniformly distributed over the entire image. Windows covering[2] an annotated object are considered positive examples ($\mathcal{W}^{\mathrm{obj}}$), the others negative ($\mathcal{W}^{\mathrm{bg}}$).

For any value of $\theta$ we can build the likelihoods for the positive $p_\theta(\mathrm{CC}(w,\theta)|\mathrm{obj})$ and negative classes $p_\theta(\mathrm{CC}(w,\theta)|\mathrm{bg})$, as histograms over the positive/negative training windows. Note how these histograms depend on $\theta$.

We now find the optimal $\theta^*$ by maximizing the posterior probability that object windows are classified as positives

$$\begin{aligned}\theta^* &= \arg\max_\theta \prod_{w \in \mathcal{W}^{\mathrm{obj}}} p_\theta(\mathrm{obj}|\mathrm{CC}(w,\theta)) = \qquad (10)\\ &= \arg\max_\theta \prod_{w \in \mathcal{W}^{\mathrm{obj}}} \frac{p_\theta(\mathrm{CC}(w,\theta)|\mathrm{obj}) \cdot p(\mathrm{obj})}{\sum_{c \in \{\mathrm{obj},\mathrm{bg}\}} p_\theta(\mathrm{CC}(w,\theta)|c) \cdot p(c)}\end{aligned}$$

where the priors are set by relative frequency: $p(\mathrm{obj}) = |\mathcal{W}^{\mathrm{obj}}|/(|\mathcal{W}^{\mathrm{obj}}| + |\mathcal{W}^{\mathrm{bg}}|)$, $p(\mathrm{bg}) = 1 - p(\mathrm{obj})$.

The advantage of this procedure is that the distribution of training samples is close to what the method will see when a new test image is presented, as opposed to just using the positive ground-truth samples and a few negative samples. Moreover, it is likely to generalize well, as it is trained from many variants of the annotated windows in $\mathcal{W}^{\mathrm{obj}}$ (i.e. all those passing the PASCAL criterion, which is also how the method will be evaluated on test images, sec. 5).

[2]We follow the widespread PASCAL criterion [17], and consider a window $w$ to cover an object $o$ if $|w \bigcap o|/|w \bigcup o| > 0.5$.

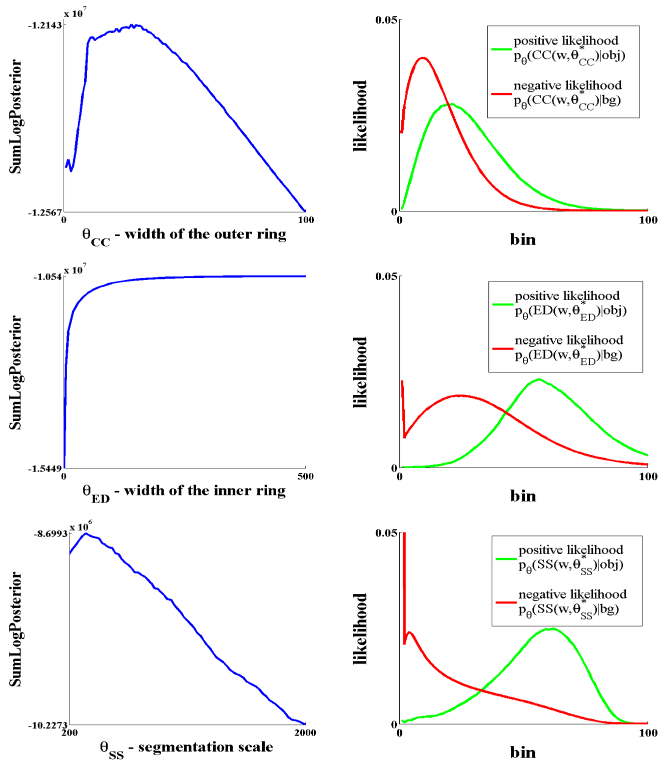


Fig. 9: **Learning parameters for CC, ED, SS**. *First column: total sum of posterior log-probabilities over all positive training examples (eq.* (10)*) as a function of* $\theta$*. We set each* $\theta^*$ *to the maximum in each case. Second column: positive (green) and negative (red) likelihoods, computed using the optimal* $\theta^*_{CC}$, $\theta^*_{ED}$, $\theta^*_{SS}$.

The learned parameters $\theta^*_{CC}$, $\theta^*_{ED}$ define the optimal outer ring $\mathrm{Surr}(w, \theta_{CC})$ and inner ring $\mathrm{Inn}(w, \theta_{ED})$. The parameter $\theta^*_{SS}$ defines the optimal superpixel segmentation scale.

As fig. 9 shows, CC, SS have a well defined optimal value $\theta^*$. For CC, the optimal size $\theta^*_{CC}$ of the outer ring captures the amount of context with maximal discriminative power. In constrast, the posterior (10) for ED continuously increases with $\theta_{ED}$, showing that the maximal discriminative power is found when computing the density of edgels over the *entire window*.

### 3.2 Learning the parameters of MS

We learn each threshold $\theta^s_{\mathrm{MS}}$ independently, by optimizing the localization accuracy of the training object windows $\mathcal{O}$ at each scale $s$. For every training image $I$ and scale $s$, we compute the saliency map $I^s_{\mathrm{MS}}$ and the MS score of all windows. We run non-maximum suppression on this 4D score space using the efficient technique of [43], resulting in a set of local maxima windows $\mathcal{W}^s_{\max}$. We find the optimal $\theta^{s*}_{\mathrm{MS}}$ by maximizing

$$\theta^{s*}_{\mathrm{MS}} = \arg\max_{\theta^s_{\mathrm{MS}}} \sum_{o \in \mathcal{O}} \max_{w \in \mathcal{W}^s_{\max}} \frac{|w \bigcap o|}{|w \bigcup o|} \qquad (11)$$

i.e. we seek for the threshold $\theta^{s*}_{\mathrm{MS}}$ that leads the local maxima of MS in the images to most accurately cover the annotated

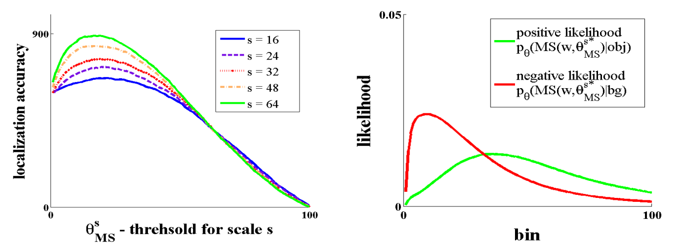


Fig. 10: **Learning parameters for MS**. *Left: localization accuracy (eq. (11)) as a function of* $\theta^s_{\text{MS}}$*, for each scale s. Right: positive (green) and negative (red) likelihoods, using the optimal* $\theta^{s*}_{MS}$*.*

objects $\mathcal{O}$. Notice how this procedure is discriminative. Maximizing (11) implicitly entails minimizing the score of windows not covering any annotated object.

### 3.3 Learning the parameters of LS

The covariance matrix $\theta_{LS}$ is considered diagonal

$$\theta_{LS} = \textbf{diag}(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \qquad (12)$$

We learn the standard deviations $\sigma_i$ using a k-nearest neighbours approach. As all four $\sigma_i$ are learned in the same manner, we restrict the explanation to $\sigma_1$.

For each training window $w_i \in \mathcal{O}$ we compute its k-nearest neighbours in the 4D Euclidian space $\mathcal{W}$, and then derive the standard deviation of the first dimension over these neighbors. We set $\sigma_1$ to the median of these standard deviations over all training windows. Fig. 8 shows marginalizations of the learned kernel density estimate of the 4D distribution over location and size dimensions. Clearly, these distributions are not uniform, which is the root of the discriminative power of the LS cue. In our experiments we use $k = 10$.

## 4 Bayesian cue integration

Since the proposed cues are complementary, using several of them at the same time appears promising. MS gives only a rough indication of where an object is as it is designed to find blob-like things (fig. 2). Instead, CC provides more accurate windows, but sometimes misses objects entirely (fig. 3). ED provides many false positives on textured areas (fig. 4). SS is very distinctive but depends on good superpixels, which are fragile for small objects (fig. 6). LS provides a location-size prior without analyzing image pixels.

To combine $n$ cues $\mathcal{C} \subseteq \{\text{MS}, \text{CC}, \text{ED}, \text{SS}, \text{LS}\}$ we train a Bayesian classifier to distinguish between positive and negative $n$-uples of values (one per cue). For each training image, we sample 100000 windows from the distribution given by the MS cue (thus biasing towards better locations), and then compute the other cues in $\mathcal{C}$ for them. Windows covering an annotated object are considered as positive examples $\mathcal{W}^{\text{obj}}$, all others are considered as negative $\mathcal{W}^{\text{bg}}$.

A natural way to combine the cues is to model them jointly. Unfortunately, integrating many cues $\mathcal{C}$ would require an enormous number of samples to estimate the joint likelihood $p(\text{cue}_1, \ldots, \text{cue}_n|\text{obj})$, where $\text{cue}_i \in \mathcal{C}$. Therefore, we choose a Naive Bayes approach. We have also tried a linear discriminant, but it performed worse in our experiments, probably because it combines cues in a too simplistic manner (i.e. a weighted sum).

In the Naive Bayes model, the cues are independent, so training consists of estimating the priors $p(\text{obj}), p(\text{bg})$, which we set by relative frequency, and the individual cue likelihoods $p(\text{cue}|\text{c})$, for cue $\in \mathcal{C}$ and $c \in \{\text{obj}, \text{bg}\}$, from the large sets of training windows $\mathcal{W}^{\text{obj}}, \mathcal{W}^{\text{bg}}$.

After training, when a test image is given, we can sample any desired number $T$ of windows from MS and then compute the other cues for them (as done above for the training windows). The posterior probability of a test window $w$ is

$$\begin{aligned} p(\text{obj}|\mathcal{C}) &= \frac{p(\mathcal{C}|\text{obj})p(\text{obj})}{p(\mathcal{C})} \qquad (13) \\ &= \frac{p(\text{obj}) \prod_{\text{cue}\in\mathcal{C}} p(\text{cue}|\text{obj})}{\sum_{c\in\{\text{obj},\text{bg}\}} p(\text{c}) \prod_{\text{cue}\in\mathcal{C}} p(\text{cue}|\text{c})} \end{aligned}$$

The posterior given by eq. (13) constitutes the final objectness score of $w$. The $T$ test windows and their scores (13) form a distribution from which we can sample any desired final number $F$ of windows. Note how eq. (13) allows us to combine any subset $\mathcal{C}$ of cues, e.g. pairs of cues $\mathcal{C} = \{\text{MS}, \text{CC}\}$, triplets $\mathcal{C} = \{\text{MS}, \text{CC}, \text{SS}\}$ or all cues $\mathcal{C} = \{\text{MS}, \text{CC}, \text{ED}, \text{SS}, \text{LS}\}$. Function (13) can combine any subset rapidly without recomputing the likelihoods.

**Multinomial sampling.** This procedure samples independently windows according to their scores. $T$ window scores form a multinomial distribution D. Naively sampling $F$ windows from D requires $T \cdot F$ operations, so we use an efficient sampling procedure. From the $T$ scores we build the cumulative sum score vector $v$. Note how the elements of $v$ are sorted in ascending order and the last vector element $v(T)$ is the sum of all scores. To sample a window we first generate a random number $u$ uniformly distributed in $[0, v(\text{T})]$. Then we do a binary search in $v$ to retrieve the interval $[v_{i-1}, v_i]$ containing $u$. The chosen sample $i$ has score $v_i$. Hence, sampling F windows only costs $\text{F} \cdot \log_2 \text{T}$ operations.

**NMS sampling.** This procedure samples windows according to their individual scores and the spatial overlap between windows. The goal is twofold: sample high scored windows *and* cover diverse image locations. This helps detecting more objects. We start by sampling the single highest scored window. Then we iteratively consider the next highest scored window and sample it if it does not overlap strongly with any higher scored window (i.e. intersection-over-union $> 0.5$). This is repeated until the desired $F$ samples are obtained.

## 5 Experiments

We evaluate the performance of the objectness measure on the popular PASCAL VOC 07 dataset [17], which is commonly used to evaluate class-specific object detectors [11, 18, 50]. We report results on the test part of the dataset, which consists of 4952 images where all object instances from twenty categories are annotated by bounding-boxes. The large number of objects and the variety of classes make this dataset best suited to our evaluation, as we want to find *all* objects in the image, irrespective of their classes.

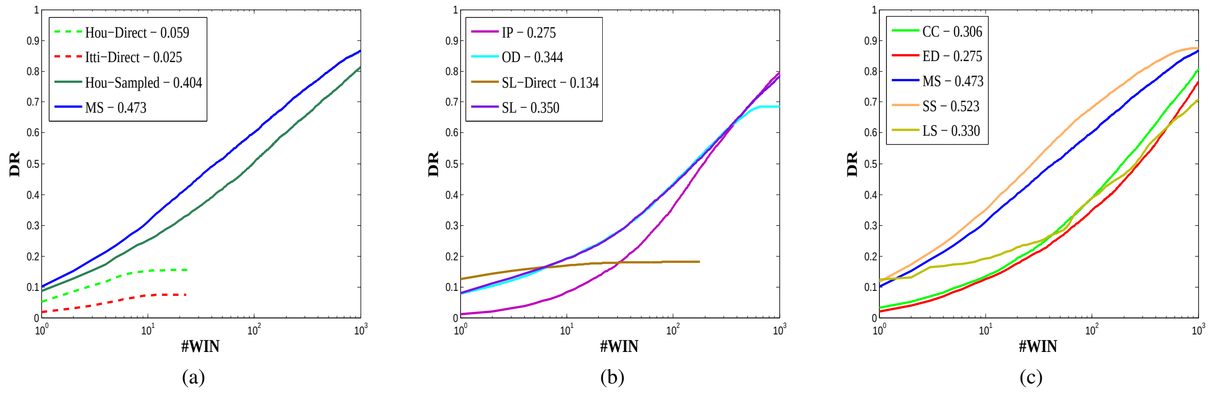


Fig. 11: **DR-#WIN plots.** *(a)* MS *vs [27, 28]; (b) baselines and semantic labeling [22]; (c) single cues.*

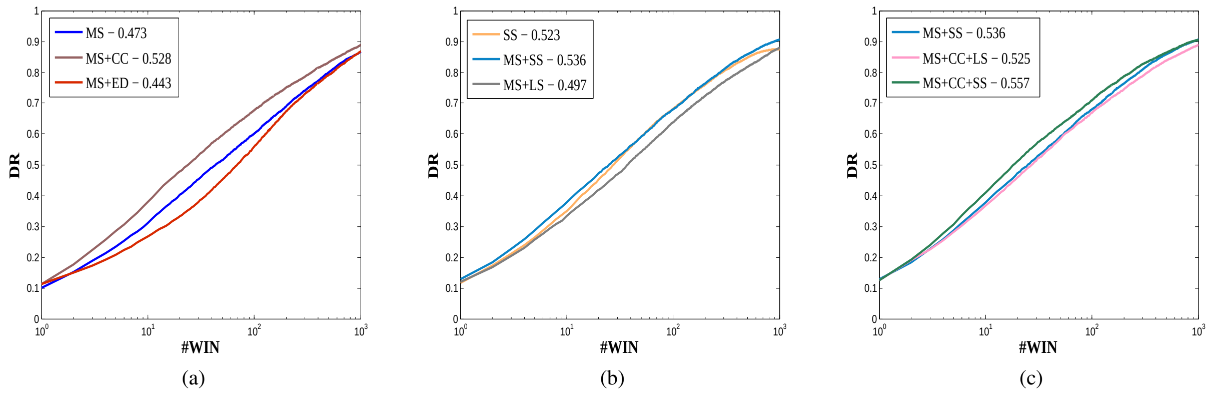


Fig. 12: **DR-#WIN plots.** *(a) two cue pair combinations vs MS; (b) another two cue pair combinations vs SS; (c) cue triplet combinations vs the best pair MS+SS.*

We demonstrate that objectness is *generic over classes* by testing on images not containing any class used for training (sec. 3). This results in a test set comprising 2941 images with 7610 object instances of the classes {*aeroplane, bicycle, boat, bottle, bus, chair, diningtable, horse, motorbike, person, pottedplant, sofa, train, tvmonitor*}. Note how these classes are of considerably different nature than the training ones. Finally, this dataset is very challenging: the objects appear against heavily cluttered backgrounds and vary greatly in location, scale, appearance, viewpoint and illumination.

**DR-#WIN curves.** Performance is evaluated with curves measuring the detection-rate vs number of windows (DR-#WIN). #WIN is the number of windows output by the algorithm being evaluated. DR is the percentage of ground-truth objects covered by those windows. An object is considered covered by a window if the strict PASCAL-overlap criterion is satisfied (intersection-over-union $> 0.5$ [17]). For comparing methods, we summarize a DR-#WIN curve with a single value: the area under the curve (AUC), after renormalizing the horizontal axis from $[0, 1000]$ to $[0, 1]$, so the AUC ranges in $[0, 1]$.

**Setup.** As there are millions of windows in an image it is expensive to compute all cues for all windows. Moreover, our measure is intended as a focus of attention mechanism for later applications, so it is desirable to output a much smaller number of windows likely to cover objects.

For ED, MS, SS, LS, for each image we score all the $T$ windows on a regular 4D grid, then sample 100000 windows using the multinomial procedure and store them in a table $D$. Computing CC is more expensive due to the large number of integral images involved (one per color bin). Therefore, for CC we build D by scoring $T = 100000$ windows sampled uniformly over the image. We compute each window score as its posterior probability to cover an object. Finally, to evaluate each cue we sample 1000 windows out of D using the NMS procedure.

For a cue combination, we build D by multinomial sampling $T = 100000$ windows from the distribution given by MS, then compute the other cues for these windows, and finally rescore them with eq. (13). This is a very efficient and accurate approximation to sampling directly from (13), as essentially all objects are covered by 100000 samples from MS. Finally, to evaluate a cue combination we consider the first 1000 windows sampled from D using the NMS procedure. Therefore, each individual cue and each cue combination is evaluated uniformly on 1000 windows per image.

**MS vs [27, 28]** We compare our MS cue to [27] and [28] (fig. 11a). The Hou-Direct curve refers to [27] as originally proposed[3]. A single saliency map at scale $s = 64$ is thresholded at its average intensity[4]. Each connected component in the resulting binary map is output as an object. The Itti-Direct

[3]software from http://www.klab.caltech.edu/∼xhou/projects/spectralResidual/
[4]this gives better results than the threshold of [27], i.e. 3× the average

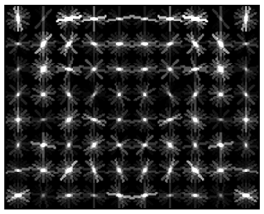


Fig. 13: **Trained HOG template**. *A visualization of a HOG model trained on all objects from the same training set $\mathcal{T}$ used to train the objectness measure (sec. 3). Brighter line segments indicate gradient directions and positions which are given higher weight by the SVM classifier [11].*

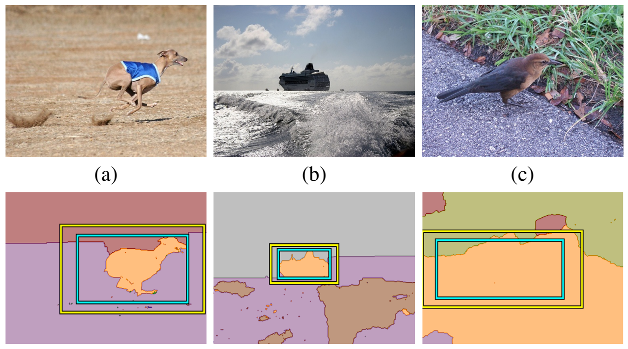


Fig. 14: **SL success and failure. Success**: *The windows covering the objects in (a) and (b) have high* SL *score, based on the semantic segmentations in (d) and (e). The percentage of foreground pixels (orange) inside the windows (cyan) is much larger than the percentage in the surrounding ring (yellow),* **Failure**: *The semantic labeling (f) failed to classify the road as background. The resulting overgrown foreground segment leads to low* SL *for the window on the bird. The surrounding ring is drawn for the optimal size $\theta^*_{SL}$ learned in sec. 3.*

curve refers to [28] as originally proposed. The saliency map is computed and the most salient objects are extracted from it using Itti's procedure [28][5].

Both procedures output only few windows per image (about 5-10). As the curves show, both methods fail on the challenging PASCAL VOC 07 dataset. Hou-Direct performs better than Itti-Direct, but still reaches only about 16% DR. For a fair comparison to our method, the Hou-Sampled curve reports performance when inserting [27] into our framework: we use their saliency map in our window score (eq. (2)) and learn the threshold $\theta_{64}$ from our training set as in eq. (11). Finally, the MS curve uses three color channels and multiple scales, with separate scale-specific learned thresholds. DR-#WIN performance curves improve steadily as we progress towards our MS cue, showing that all the components we propose contribute to its success.

**Single cues vs baselines.** We compare our objectness cues to two baselines (fig 11b): interest points (IP) and a *Histogram of Oriented Gradients* (HOG) [11] detector (OD).

For IP we extract laplacian based multi-scale Harris interest points [41] and score every window $w$ in a 4D regular grid by the density of IPs it contains

$$\mathrm{IP}(w) = \frac{\sum_{\{p \in w\}} \mathrm{strength}(p)}{\sqrt{|w|}} \tag{14}$$

where the summation runs over all IPs inside $w$, and $\mathrm{strength}(p)$ is the response of the IP detector. For evaluation we sample 100000 windows from the grid with multinomial sampling and then retain the first 1000 windows sampled with NMS from them. HOG was shown to perform well for class-specific object detection. Here we train it to detect objects of arbitrary classes: for OD we train a single HOG detector (fig. 13) from all object instances used to train objectness (sec. 3). For a test image, we score all windows on a dense grid based on their HOG response, then sample 100000 windows with multinomial sampling, and finally apply NMS sampling to obtain 1000 windows. As fig. 11b shows, OD performs better than IP, showing it conveys better information about objects. However, its absolute performance is considerably lower than MS. We also tried running OD at multiple aspect-ratios, and we tried using [18], which extends HOG with parts. They did not perfom better than plain HOG on this task.

Fig. 11c reports performance for our single cues. All our cues perform better than the IP baseline, which shows that finding entire objects cannot simply be reduced to interest point detection. Moreover, MS and SS considerably outperform the best baseline OD. This confirms that generic object detection is different from class-specific detection. We believe that no single pattern of gradients (fig. 13) within a window is characteristic for objects in general, whereas our cues are designed to capture this.

Our newly proposed SS cue performs best, significantly above the second-best cue MS (and therefore also above [27, 28]). This demonstrates SS is a powerful alternative to traditional 'stand out' saliency cues. Somewhat surprisingly, LS also performs quite well, about in the range of CC and OD. This shows that, even in a realistic dataset such as PASCAL VOC 07, there are locations and sizes at which objects are more likely to appear. ED is the worst performing cue, as it is strongly attracted to regions of high edgel density caused by background clutter.

**Cue combinations.** Combining cues in our Bayesian framework improve results for most combinations (fig. 12). All cue pairs combinations, but that including ED, improve over the performance of the best cue in the pair (fig. 12a and fig. 12b). Because of ED's low performance, we exclude it from more complex combinations. Adding a third cue further raises AUC for most pairs (fig. 12c): adding SS to MS+CC, or adding CC to MS+SS, leads to the best performing triplet MS+CC+SS.

The results show these three cues are complementary and all important for finding objects in highly challenging images. In contrast, while LS helps when paired with MS, it makes no positive contribution to any cue triplet. Starting from two image cues, objectness is rich enough that it no longer benefits from the addition of the pure prior LS offers. ED performs poorly in general and measures a characteristic already captured by the better SS.

Table 1 shows the DR of several cues and combinations

[5]software from http://www.saliencytoolbox.net/

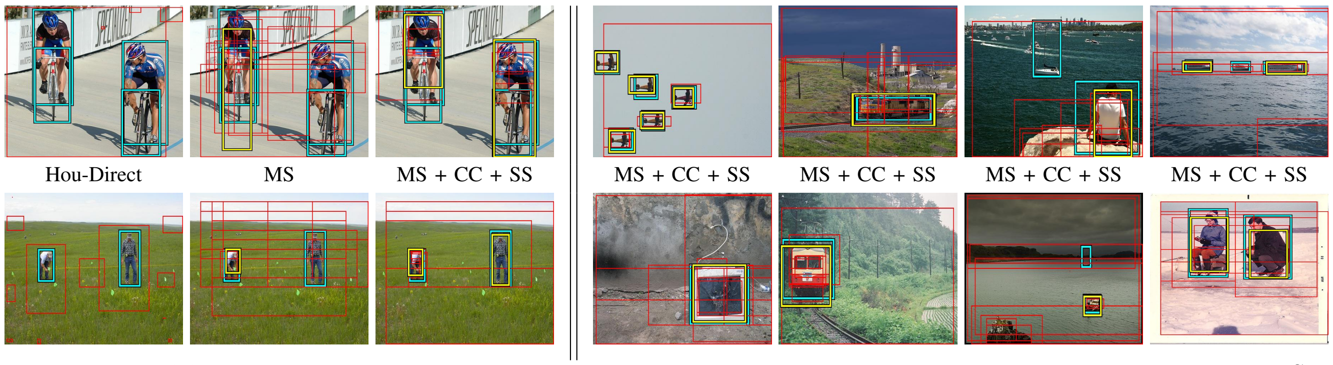


Fig. 15: **Pascal VOC 07 examples.** *First 3 columns: example output of Hou-Direct, and the first* 10 *windows sampled from* MS *and* MS + CC + SS. *We mark in yellow windows correctly covering ground-truth object (cyan); if there is more than one correct window, the best one is shown; all other windows are in red. The windows output by Hou-Direct rarely correctly cover an object.* $MS$ *finds some of the objects but misses others, while* $MS + SS + CC$ *finds more objects and its windows also cover the objects more accurately. The last four columns show the first* 10 *windows sampled from our final objectness measure* MS + CC + SS. *Even with just* 10 *windows, it can find several objects in these challenging images.*

TABLE 1: Detection rate as a function of the number of windows sampled for various cues and combinations.

| #WIN | OD | MS | SS | MS + SS | MS + CC + SS |
|---|---|---|---|---|---|
| 10 | 0.19 | 0.31 | 0.35 | 0.38 | 0.41 |
| 100 | 0.44 | 0.60 | 0.68 | 0.68 | 0.71 |
| 1000 | 0.69 | 0.87 | 0.88 | 0.91 | 0.91 |

when sampling $10, 100$ or a $1000$ windows. The best combination MS+CC+SS already detects 71% of all objects with just 100 windows per image. Moreover, it reaches 91% detection-rate with $1000$ windows per image, which are orders of magnitude fewer than all windows in the image. This high recall at such small number of sampled windows is key to the usefulness of objectness in applications. In comparison, the class-specific technique OD [11] only detects 44% and 69% of the objects with 100 and 1000 windows respectively.

In addition to high recall, MS+CC+SS also localizes objects accurately. We measure accuracy as the average area of intersection-over-union of the single window best covering each detected object, out of the $1000$ windows per image returned by the algorithm. The accuracy for MS+CC+SS is $70\%$. In general, sampling more windows leads to more detected objects but also increases the accuracy of the best available window (fig. 15, fig. 16).

**Computation times.** The time needed to compute the objectness measure of an image depends on several factors: the number of pixels in the image, which cues are combined, and the number of windows scored on the $4$D grid used to obtain the distribution $\mathrm{D}$. In PASCAL VOC 07 the average image size is $380 \times 470$ pixels. In the following we analyze the computational times for an average image.

As explained in the *Setup* paragraph, for a cue combination, the distribution $\mathrm{D}$ is built by sampling $T = 100000$ windows from the distribution given by $\mathrm{MS}$, then computing the other cues for them and finally scoring them with eq. (13). The distribution given by $\mathrm{MS}$ is obtained by scoring every window for $5$ scales and $3$ channels, for a total of about $11$ million windows, which takes about $1.5$ seconds. Sampling $T$ windows takes negligible time ($< 0.01$ seconds) for the multinomial sampling and around $0.5$ seconds for the NMS sampling. Scoring the $T$ sampled windows with the other cues costs: **(CC)**: about $1$ second, depending on the quantization of the LAB space. **(ED)**: about $0.2$ seconds, including the time to compute the edge map using the Canny detector [8]. **(SS)**: around $1.5$ seconds, including the time to compute superpixels using [19]. **(LS)**: around $0.1$ seconds given the precomputed lookup table from sec. $2.5$.

The cue combination MS+CC+SS offers the highest AUC and takes only $4$ seconds to compute (on average per image). Because of this we consider **MS+CC+SS** as the **objectness measure** and use it to speed up class-specific object detectors by greatly reducing the number of windows they evaluate (sec. 6), or to reduce their false-positive rate (sec. 7).

**Generalization over classes.** We perform here an additional experiment to further demonstrate the class-generic nature of objectness, by showing that its test performance does not depend on the classes observed during training. For this we retrain the objectness measure from a different training set, composed of 50 images randomly sampled from a variety of existing datasets, including INRIA Person, Pascal VOC 06 and Caltech 101, showing a total of 291 object instances. These are the images used in the first version of our work [3] and correspond to our software release.

We evaluate the retrained objectness measure on the same test set as in the previous paragraphs. Remarkably, the DR-#WIN performance curve remains essentially identical, changing by less than $1\%$ detection-rate at any number of windows compared to the MS+CC+SS curve in fig. 12c.

**Semantic Labeling [22].** We compare our objectness measure to the *Semantic Labeling* technique of [22] (SL). It was designed for segmenting an image into several background classes and one generic foreground class. In our experiments we use the source code[6] provided for SL within [23]. The strength of this technique lies in employing classifiers specialized to individual background classes (e.g. grass, road), which is interesting because there are only a few common background classes and each has limited appearance variabil-

[6]http://ai.stanford.edu/˜sgould/svl/

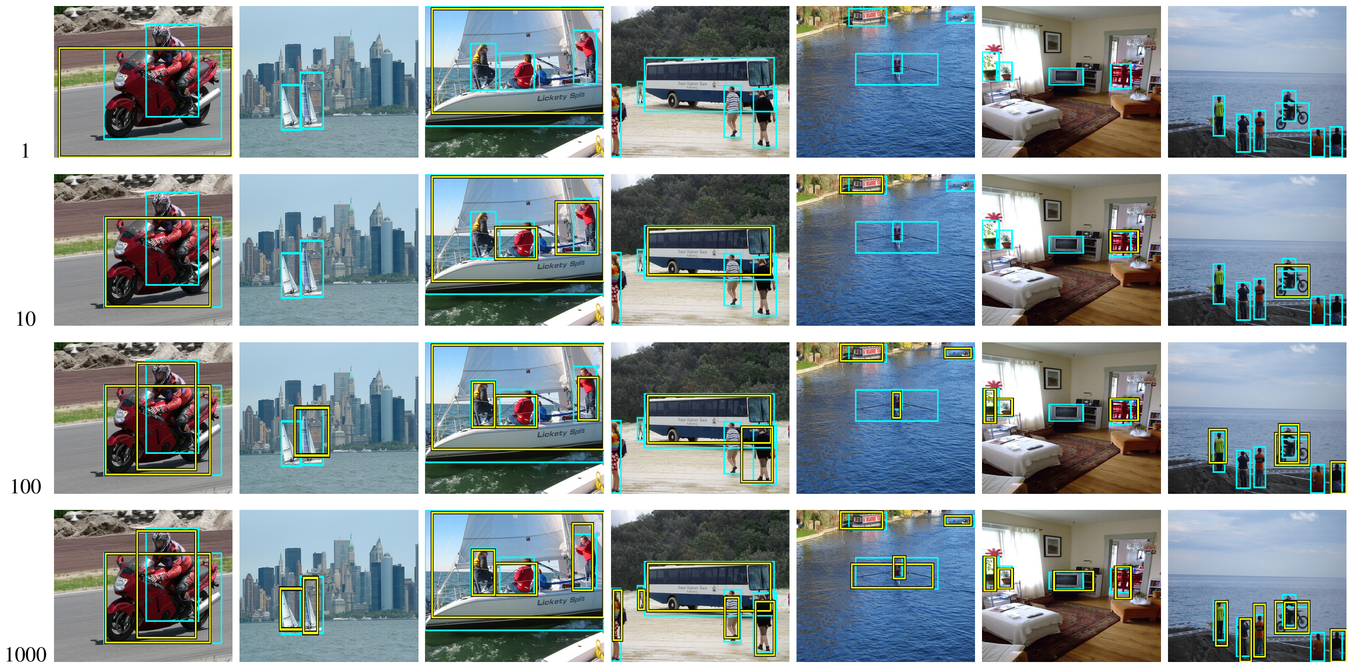


Fig. 16: **Qualitative results for increasing number of windows.** *We show the accuracy of the objectness measure as a function of the number of windows it returns. Each row shows the results obtained when sampling 1, 10, 100, and 1000 windows. We mark in yellow windows correctly covering a ground-truth object (cyan); if there is more than one correct window, the most overlapping one is shown. As we sample more windows, more objects are detected and the accuracy of the windows covering them continuously increases. Sampling 1000 windows covers nearly all objects with high accuracy. Notice that 1000 is several orders of magnitue smaller than the number of all windows in the image.*

ity. It can be seen as a complementary strategy to ours, which focuses on properties of objects.

The semantic pixel labeler of [22] divides the image into background and foreground regions. In a first experiment we directly fit a window on each foreground region and return them all as objects. Analog to Hou-Direct and Itti-Direct, this procedure makes hard decisions and returns a moderate number of disjoint windows. While it performs better than both Hou-Direct and Itti-Direct, overall it detects less than 20% of all objects.

In a second experiment, we insert [22] into our framework, in a manner analoguous to the CC cue. We compute the score $\mathrm{SL}(w, \mathrm{Surr}(w, \theta_{SL}))$ of a window $w$ as the difference between the percentage of foreground pixels inside $w$ and the percentage of foreground pixels in its immediate surrounding $\mathrm{Surr}(w, \theta_{SL})$, truncated to positive values

$$\mathrm{SL}(w, \theta_{SL}) = \max(0,\ pFG(w) - pFG(\mathrm{Surr}(w, \theta_{SL}))) \tag{15}$$

where

$$pFG(w) = \frac{|\{p \in w \mid label(p) = fg\}|}{|w|} \tag{16}$$

and $\mathrm{Surr}(w, \theta_{SL})$ is defined as in sec. 2.2. This score is highest when $w$ fits a blob of foreground pixels tightly (fig. 14e). The truncation is useful to indicate that windows with foreground density larger outside than inside have 0 probability of containing an object. The ring size $\theta_{SL}$ is learned as $\theta_{CC}$ in sec. 3.

As fig. 11b shows, SL performs quite well and it reaches a higher AUC than the baselines and the CC, ED cues. This confirms that the technique of [22] is well suited as an objectness cue, and that sampling a larger set of windows per image is important for reaching high recall. The DR-#WIN curve of SL follows a similar trend to OD. Our MS and SS cues perform better, and our best combination MS+CC+SS reaches a substantially higher AUC. On the other hand, SL localizes objects up to their outlines, whereas objectness only returns rectangular windows. Finally, we note that computing SL costs about 35 seconds, compared to only 4 seconds for objectness.

**Object proposals based on segmentation [9, 16].** We compare our objectness measure to the two recent works of [9, 16] that produce segmentations as object candidates. Both methods generate a ranked list of segments with the desired goal of covering all objects in an image. In our experiments we use the official source code provided by the respective authors [7] [8]. In order to compare to our method, we fit a bounding-box on each segment returned by the algorithms.

In fig. 17 we plot the performance of [9, 16] against our objectness measure (MS+CC+SS). As the plot shows, [9] and objectness exhibit the same perfomance when considering a rather low number of windows (up to 100). For a larger number of windows objectness has a moderate advantage, detecting 6% more objects with a 1000 windows. The method of [16] achieves moderately higher recall than objectness until up to 500 windows, while in the range 500-1000 windows objectness finds more objects. Note how both [9, 16] produce a more detailed output than objectness, offering pixelwise

[7]http://sminchisescu.ins.uni-bonn.de/code/cpmc/
[8]http://vision.cs.uiuc.edu/proposals/

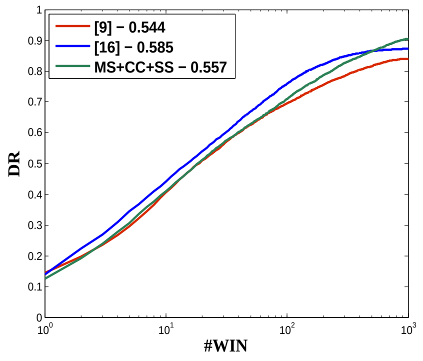


Fig. 17: **Comparison to [9, 16].** *MS+CC+SS vs [9, 16].*

```
Algorithm 1 Using objectness for class-specific detectors.
Input: $F, D, c$
Output: $\mathcal{D}et$
1: $\mathcal{I} = \{w_1, \ldots, w_F\}$, $w_i \rightsquigarrow D, \forall i$
2: $\mathcal{I}_s = \{(w_1, s_{w_1}), \ldots, (w_F, s_{w_F})\}$, $s_{w_i} = c(w_i), \forall i$
3: $\mathcal{P}_s = NMS(\mathcal{I}_s) = \{(w_{n_1}, s_{w_{n_1}}), \ldots, (w_{n_P}, s_{w_{n_P}})\}$
4: $\mathcal{L} = \{w_{n_1}^{lm}, \ldots, w_{n_P}^{lm}\}$, $w_{n_j}^{lm} = \max_{w \in \mathcal{V}_{w_{n_j}}} s_w$
5: $\mathcal{D}et = NMS(\mathcal{L})$
```

segmentations instead of rectangular windows. In terms of computation, both [9, 16] are very expensive (about 420 seconds per image for [9], 120 for [16]). Depending on the application, this might be prohibitively slow (e.g. sec. 6).

## 6 Speeding up class-specific detectors

Many modern class-specific object detectors [12, 18, 25] are based on sliding windows. In sec. 6.1 we give a general algorithm for using our objectness measure as a location prior for *any* sliding window detector. In sec. 6.2-6.4 we detail how this algorithm works for three particular detectors [11, 18, 33]. We show in sec. 6.5 that objectness greatly reduces the number of windows evaluated by the detectors.

### 6.1 General algorithm

The general scheme for using our objectness measure as a location prior for object detectors is algorithm 1. The algorithm inputs the class-specific confidence function $c$ which the detector employs to score a window.

We build an initial set $\mathcal{I}$ of $\mathrm{F} = 1000$ windows multinomially sampled from the distribution D of windows scored by our objectness measure $\mathrm{MS} + \mathrm{CC} + \mathrm{SS}$ (line 1). We use $c$ to score each window in $\mathcal{I}$ (line 2). We then run the non-maxima suppression of [18]. This results in a set $\mathcal{P}_s$ of promising windows (line 3). For every window $w_p \in \mathcal{P}_s$, we iteratively move to the local maximum of $c$ in its neighborhood $\mathcal{V}_{w_p}$, resulting in window $w_p^{lm}$ (line 4). Finally, we run NMS on the local maxima windows $\mathcal{L}$ and obtain detections $\mathcal{D}et$ (line 5).

In order to use this algorithm one has to specify a window scoring function $c$, which is specific to a particular detector and object class, and a window neighborhood $\mathcal{V}$.

TABLE 2: For each detector [11, 18, 33] we report its performance (left column) and that of our algorithm 1 using the same window scoring function (right column). We show the average number of windows evaluated per image #win and the detection performance as the mean average precision (mAP) over all 20 classes.

| | [11] | OBJ- [11] | [18] | OBJ- [18] | ESS-BOW[33] | OBJ-BOW |
|---|---|---|---|---|---|---|
| mAP | 0.186 | 0.162 | 0.268 | 0.225 | 0.127 | 0.125 |
| #win | 79945 | 1349 | 18562 | 1358 | 183501 | 2997 |

### 6.2 Speeding up [11]

The detector of [11] models a class by a single window filter $c$ on HOG features. In [11] this filter is applied at all positions $(x, y)$ and scales $s$ on a grid over image. In our algorithm instead, we map a window $w_i \in \mathcal{I}$ to the best corresponding location $(x,\ y,\ s)$ using the filter's aspect-ratio, and then score it with $c$. Next, we search for a local maximum in the neighborhood $\mathcal{V}_{w_i}$ on the grid, for up to 5 iterations. We use the neighborhood $\mathcal{V}_{w_i} = \{x-1, x, x+1\} \times \{y-1, y, y+1\} \times \{s-1, s, s+1\}$ comprising 27 windows.

### 6.3 Speeding up [18]

The detector of [18] models a class with a mixtures of multiscale deformable part models. More complex than [11], the model includes a root filter and a collection of part filters and associated deformation models. The score of a window at location $(x, y, s)$ combines the score of the root filter, the parts and a deformation cost measuring the deviation of the part from its ideal location.

In our algorithm, we map a window $w_i \in \mathcal{I}$ to a location $(x, y, s)$ using the root's aspect ratio, and then score it with $c$. We use the same neighborhood and number of local search iterations as in sec. 6.2.

### 6.4 Speeding up Bag-of-words detectors

The window scoring function $c$ can also be a Bag-of-visual-words classifier [33], which represents a window with a histogram of local features quantized over a precomputed codebook. Here we follow the setup of [33] and use a linear SVM as $c$ .

No mapping to a location $(x, y, s)$ is necessary, as the BOW descriptor is defined on windows of any aspect-ratio. We score only the sampled windows $w_i \in \mathcal{I}$ and set the neighborhood $\mathcal{V}_{w_i}$ to all windows obtained by translating $w_i$ by 2 pixels and/or scaling $w_i$ by factor 1.1. We iterate local search until convergence (usually about 10 iterations).

Since for this type of window scoring function it is possible to apply the branch-and-bound technique ESS [33], we compare to ESS rather than to traditional sliding-windows on a grid [11, 18].

### 6.5 Quantitative evaluation

We compare the 3 object detectors [11, 18, 33] to our algorithm 1 on the entire PASCAL VOC 07 test set (20 object classes over 4952 images) using the standard PASCAL VOC protocol [17].

Our algorithm uses the same window scoring function $c$ used by the corresponding detector. We train it for [18] using

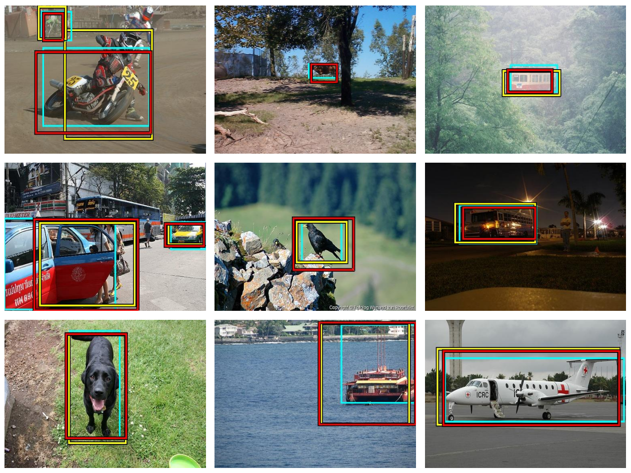


Fig. 18: **Class-specific detections.** *We show the output of three object detectors in yellow (first row - [11], second row - [18], third row -* ESS*). We show in red the output of our algorithm using the same window scoring function as the corresponding detector. Ground-truth objects are in cyan.*

their source code[9]. For [11] we train [18] with only the root filter (no parts). For [33] we obtained precomputed image features and SVM hyperplanes from the authors. For detection, we use the source code of [18] for both [18] and [11], and the source code of ESS[10] for [33].

As tab. 2 shows, our algorithm evaluates 15×-60× fewer windows than the sliding-windows approaches [11, 18]. Interestingly, it also evaluates 60× fewer windows than ESS [33] (notice that the implicit search space of [33] is larger than that of [11, 18] as it contains all windows, not just a grid). Moreover, our algorithm's mAP is only slightly below the original detectors ($-0.023$ on average), showing this massive speedup comes at little compromise on performance. The additional cost to compute objectness and sample 1000 windows $\mathcal{I}$ is negligible as it is done only once per image. The same windows $\mathcal{I}$ are reused for all 20 classes, and can in fact be reused for *any* class. For example, [18] takes about 4 seconds for each class, while objectness takes about 4 seconds only once. Finally, our algorithm is general, as it supports *any* window scoring function. ESS instead is restricted to functions for which the user can provide a (tight) bound on the best score in a contiguous set of windows (for example, for the scoring functions of [11, 18] no bound is known and ESS is not applicable).

## 7 Reducing false-positive rates of class-specific detectors

The objectness measure can also be used to reduce the number of false-positives returned by class-specific object detectors. In the following we explain how to use objectness as a complementary score in addition to the class-specific scoring function of any detector. We demonstrate experimentally that this combination leads to fewer false positives for the detector [18].

[9]http://people.cs.uchicago.edu/~pff/latent-release3/

[10]http://sites.google.com/a/christoph-lampert.com/work/software

**Adding objectness to a class-specific detector.** The class-specific scoring function of a detector typically returns a high response to instances of the class, but occasionally also to other image patterns, leading to false-positive detections. Some of these false-positives cover background patterns or are partially on an object and on background. On these windows we expect objectness to give a low score.

Following this intuition we linearly combine the class-specific score $c(w)$ of a window $w$ with its objectness score $p(obj|w)$ (eq. 13): $c(w) + \alpha \cdot p(obj|w)$. This combination can be used instead of the class-specific score inside the innermost loop of any sliding window detector. All other elements of the standard detector pipeline remain the same (e.g. NMS). The weight $\alpha$ controls the importance of the objectness score.

**Quantitative evaluation for [18].** We present results for adding objectness to the pipeline of [18]. We set $\alpha = 0.2$. In table 3 we show the results obtained on the test set of PASCAL VOC 07 (all 20 classes) [11]. As table 3 shows, adding objectness improves the mAP for 11 classes. Although the improvement is not large, it is significant given the size of this dataset. Moreover, note how [18] is a high performance, state-of-the-art algorithm, which is difficult to further improve upon. Therefore, we consider the increase in mAP brought by adding objectness quite exciting (about 1% on average over all classes).

## 8 Conclusions

We presented an objectness measure trained to distinguish object windows from background ones. It combines several image cues, including the innovative SS cue. On the task of detecting objects of new classes unseen during training, we have demonstrated that objectness outperforms traditional saliency [27, 28], interest point detectors, Semantic Labeling [22], and the HOG detector [11]. Moreover, we have demonstrated algorithms to employ objectness to greatly reduce the number of windows evaluated by class-specific detectors, and to reduce their false-positive rates. Several recent works are increasingly demonstrating the value of objectness in other applications, such as learning object classes in weakly supervised scenarios [13, 30, 47], pixel-wise segmentation of objects [2, 52], unsupervised object discovery [34], and learning humans-object interactions [45]. The source code of the objectness measure is available at http://www.vision.ee.ethz.ch/~calvin.

**Acknowledgements.** We thank A. Zisserman for suggesting the basic intuition behind the LS cue and D. Koller and S. Gould for helping us using their Semantic Labeling software.

## References

[1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequancy-tuned salient region detection. In *CVPR*, 2009.
[2] B. Alexe, T. Deselaers, and V. Ferrari. ClassCut for unsupervised class segmentation. In *ECCV*, 2010.
[3] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010.
[4] J. Arpit, R. Saiprasad, and M. Anurag. Multi-stage contour based detection of deformable objects. In *ECCV*, 2008.

[11]no image used to train objectness is in this test set (sec. 3).

TABLE 3: We report the performance of [18] alone (first row) and when combined with objectness (second row). We show the detection performance as the average precision (AP) for the 20 classes in the PASCAL VOC 07 test set.

| | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbik | pers | plant | sheep | sofa | train | tv | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [18] | .286 | .551 | .006 | .145 | .265 | .397 | .501 | .165 | .165 | .168 | .246 | .050 | .452 | .383 | .358 | .090 | .174 | .227 | .341 | .383 | **0.268** |
| our | .286 | .545 | .011 | .142 | .267 | .420 | .502 | .182 | .165 | .175 | .262 | .077 | .468 | .396 | .362 | .116 | .141 | .231 | .348 | .392 | **0.274** |

[5] X. Bao, T. Narayan, A. A. Sani, W. Richter, R. R. Choudhury, L. Zhong, and M. Satyanarayanan. The case for context-aware compression. In *ACM Hotmobile*, 2011.

[6] T. L. Berg and A. Berg. Finding iconic images. In *CVPR Internet Vision Workshop*, 2009.

[7] N. Bruce and J. Tsotsos. Saliency based on information maximization. In *NIPS*, 2005.

[8] J. F. Canny. A computational approach to edge detection. *IEEE Trans. on PAMI*, 8(6):679–698, 1986.

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

[10] F. Crow. Summed-area tables for texture mapping. In *SIGGRAPH*, 1984.

[11] N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In *CVPR*, volume 2, pages 886–893, 2005.

[12] C. Desai, D. Ramanan, and C. Folkess. Discriminative models for multi-class object layout. In *ICCV*, 2009.

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

[14] R. Desimone and J. Duncan. Neural mechanisms of selective visual-attention. *Ann. Rev. Neurosci.*, 1(18):193–222, 1995.

[15] W. Einhauser and P. Konig. Does luminance-contrast contribute to saliency map for overt visual attention. *European Journal of Neuro-science*, 5(17):1089–1097, 2003.

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

[17] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 Results, 2007.

[18] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 32(9):1627–1645, 2010.

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

[20] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. on PAMI*, 30(1), 2008.

[21] D. Gao and N. Vasconcelos. Bottom-up saliency is a discriminant process. In *ICCV*, 2007.

[22] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009.

[23] S. Gould, O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Ng, and D. Koller. *The STAIR Vision Library (v2.4)*, 2010. Software available at http://ai.stanford.edu/ sgould/svl.

[24] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *NIPS*, 2007.

[25] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[26] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *ECCV*, 2008.

[27] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *CVPR*, 2007.

[28] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *pami*, 20(11):1254–1259, 1998.

[29] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *ECCV*. Springer-Verlag, May 2004.

[30] I. Khan, P. M. Roth, and H. Bischof. Learning object detectors from weakly-labeled internet images. In *OAGM Workshop*, 2011.

[31] W. Kienzle, F. Wichmann, A., B. Scholkopf, and M. Franz, O. A nonparametric approach to bottom-up visual saliency. In *NIPS*, 2006.

[32] G. Krieger, I. Rentschler, G. Hauske, K. Schill, and C. Zetzsche. Object and scene analysis by saccadic eye-movements: an investigation with higher-order statistics. *Spatial Vision*, 2(16):201–214, 2000.

[33] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

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

[35] B. Leibe and B. Schiele. Scale-invariant object categorization using a scale-adaptive mean-shift search. In *DAGM04*, 2004.

[36] T. Liu, J. Sun, N. Zheng, X. Tang, and H. Shum. Learning to detect a salient object. In *CVPR*, 2007.

[37] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, Sep 1999.

[38] Y. F. Ma and H. J. Zhang. Contrast-based image attention analysis by using fuzzy growing. In *ICMM*, 2003.

[39] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.

[40] L. Marchesotti, C. Cifarelli, and G. Csurka. A framework for visual saliency detection with applications to image thumbnailing. In *ICCV*, 2009.

[41] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 1(60):63–86, 2004.

[42] F. Moosmann, D. Larlus, and F. Jurie. Learning saliency maps for object categorization. In *ECCV*, 2006.

[43] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *Proc. ICPR*, 2006.

[44] E. Parzen. On the estimation of a probability density function. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[45] A. Prest, C. Schmid, and V. Ferrari. Weakly supervised learning of interactions between humans and objects. *TPAMI (accepted for publication, to appear)*, 2011.

[46] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

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

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

[49] R. Valenti, N. Sebe, and T. Gevers. Image saliency by isocentric curvedness and color. In *ICCV*, 2009.

[50] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

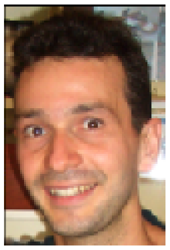
[51] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[52] A. Vezhnevets, V. Ferrari, and J. M. Buhmann. Weakly supervised semantic segmentation with multi image model. In *ICCV*, 2011.

[53] G. Wang and D. Forsyth. Joint learning of visual attributes, object classes and visual saliency. In *ICCV*, 2009.

**Bogdan Alexe** received the diploma in Mathematics and Computer Science (2005) and the MSc in Computer Science (2007), both from the University of Bucharest. Since 2008 he is a PhD student in computer vision at ETH Zurich. His research interest include machine learning and pattern recognition in computer vision, e.g. object classification, detection and recognition but also data mining and optimization methods.

**Thomas Deselaers** is software engineer at Google Switzerland in Zurich. He received his PhD degree from RWTH Aachen University in 2004 and has been a post-doctoral researcher at ETH Zurich in 2009-2010. From March 2004 to December 2008 he was a full time researcher at the Human Language Technology and Pattern Recognition Group of the Computer Science Department of RWTH Aachen University, where he has been the head of the image processing and understanding group since 2005.



**Vittorio Ferrari** is Assistant Professor at ETH Zurich. He received his PhD from ETHZ in 2004 and has been a post-doctoral researcher at INRIA Grenoble and the University of Oxford. In 2008 he was awarded a Swiss National Science Foundation Professorship grant for outstanding young researchers. He is the author of fifty technical publications, most of them in the highest ranked conferences and journals in computer vision and machine learning.