Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 164

Efficient Subwindow Search for Object Localization

Matthew B. Blaschko[1] Thomas Hofmann[2]
Christoph H. Lampert[1]

August 2007

[1] Department Schölkopf, email: chl;blaschko@tuebingen.mpg.de
[2]Google Inc., Zurich, Switzerland, email: thofmann@google.com

This report is available in PDF–format at http://www.kyb.tuebingen.mpg.de/techreports.html

Efficient Subwindow Search for Object Localization

Matthew B. Blaschko *Thomas Hofmann* *Christoph H. Lampert*

Abstract. Recent years have seen huge advances in object recognition from images. Recognition rates beyond 95% are the rule rather than the exception on many datasets. However, most state-of-the-art methods can only decide if an object is present or not. They are not able to provide information on the object location or extent within in the image.

We report on a simple yet powerful scheme that extends many existing recognition methods to also perform localization of object bounding boxes. This is achieved by maximizing the classification score over all possible subrectangles in the image. Despite the impression that this would be computationally intractable, we show that in many situations efficient algorithms exist which solve a generalized maximum subrectangle problem.

We show how our method is applicable to a variety object detection frameworks and demonstrate its performance by applying it to the popular *bag of visual words* model, achieving competitive results on the PASCAL VOC 2006 dataset.

---

1 Introduction

Determining the exact position of an object within an image is much more difficult than simply saying whether there is an object present at all. One has to recognize all parts of an object instead of relying on just the most discriminative ones. Also, it is no longer possible to rely on the fact that for many datasets, the image background is strongly correlated with the presence of certain objects. Perhaps this is the reason why there has been significantly less work done on localization than on classification. In the recent PASCAL Visual Object Classes Challenge 2006 (VOC2006) [1] there were 23 submissions for the classification task, but only 9 submissions for the localization task, and only 2 of these submitted results for all object classes. This indicates that much of the human effort in developing classification methods is not used to also improve localization. We aim to help bridge this gap by providing a general method to convert classification algorithms into localization algorithms.

The precise definition of object localization varies in the literature. Some techniques define object localization to be the identification of an object perimeter [2], others the object center [3]. Many approaches build hierarchical parts models, giving an estimate of the object center as well as its constituent parts [4, 5, 6, 7]. Conditional random fields have been applied to compute a spatially coherent estimate of whether part of an object is present in rectangular patches of the image [8]. Recently, Russell et al. have used multiple segmentations and topic models to extract class specific segments from the image in an unsupervised setting [9]. Another common approach is to provide a map of the image plane that codes how likely an object is to be present in a specific pixel, but to not explicitly specify where an object is, or if there is more than one object present [10]. We have chosen here to define object localization as the placement of a bounding box around the object of interest. This is the most common parametric description of object locations, and has been used in settings such as VOC2006 [1] due to its intuitive handling and the relative ease of providing ground truth as compared to providing complete segmentation data.

Object localization using bounding boxes has been performed using several approaches in the literature. The kind of search used depends on which model is employed, but we are not aware of previous work that has optimally maximized a score over the entire space of subwindows. Sliding window approaches have been used extensively for localization [11, 12], but due to the expense of evaluating each location and taking the maximum, only a fixed subset of windows are typically used. Rather than improving the search strategy over candidate windows, much work has been done to build an efficient classifier so that the cost per window evaluation is decreased [12]. Local greedy search has been used with an MRF model to find the extent of man-made structures in satellite imagery [13]. An initial location is provided, and the extent is iteratively found by growing a rectangular region. Bouveyron et al. model pixel appearance using a Gaussian mixture model, and then determine object localization

by computing the mean and variance of the point coordinates weighted by their posterior probability of belonging to an object [14]. These estimated means and variances are used heuristically as the subrectangle center and extent respectively. Another iterative approach is [15], where an initial object region is estimated as the bounding box of a certain number of the most discriminant feature locations in an image, and this is iteratively improved by a gradient-decent based procedure.

2 Maximal Subwindow Localization (MSL)

In contrast to previous methods, we provide a technique that efficiently finds the *optimal* bounding box according to a classification score, and that is applicable to a wide variety of existing classification systems.

The underlying intuition of the proposed method is that a classification function that is constructed to take high values for images with a certain object present, should have its highest values when applied to images which only contain the object and no background clutter. Applied to subimages of a single image, this means that the classification score is highest on a subimage showing exactly the object. Inverting this argument, the location and extent of an object to be detected is given by the image subrectangle on which the classification function takes its maximal value.

Finding an object by evaluating the classification function naively on all subimages is doomed to failure, since there are $O(n^2m^2)$ subrectangles in an $n \times m$ image. Instead, heuristic optimization procedures have been proposed e.g. PRIM [16], which relies on a greedy shrinking process followed by region growing. However, we will show in Sections 3 and 6 that for several existing classification systems, the problem can be transformed into the simpler problem of finding the subrectangle of a matrix that has the maximum sum of entries, known as the *maximum sum subrectangle problem*. This classical problem has an intuitive $O(n^2m)$ algorithm to find the optimal solution, which we will explain in Section 4.1. In recent years some algorithms with lower complexity bound have been developed, e.g. $O(n^2m(\log\log m/\log m)^{1/2})$ [17, 18].

Additionally, we will give a branch-and-bound algorithm in Section 4.2 that solves the same problem and—even though its worst case complexity is quartic—in practice runs in linear to quadratic time on image datasets. This algorithm also allows us to solve the problem for more general quality functions than the maximum sum of entries.

3 Localization for Bag of Visual Words Classifiers

We begin by demonstrating the proposed method of turning a classifier into a localization routine using a simple example, the *bag of visual words (bovw)* model. We make use of the established methods from that field, for more details please refer the vast literature on object recognition using *bovw* representations, e.g. [19, 20, 21, 22, 23, 24].

3.1 Classification using Bag of Visual Words

To a given set of training images $I^1, \ldots, I^N$ with binary class labels $y^1, \ldots, y^N$, we apply a salient point and feature detector such as SIFT [25], obtaining keypoint locations x_j^i with descriptors d_j^i. We build a codebook of representative image descriptors by applying K-means clustering to the set of descriptors in the training set. For each x_j^i, we store the discrete label c_j^i of the codebook entry closest to d_j^i.

We represent images by their cluster histograms, h^i i. e. we count how often each cluster label occurs in the image. On the histograms we train a support-vector machine with linear kernel [26]. To classify whether a new image I' contains an object or not, we build its cluster histogram h' and decide based on the value of the SVM decision function on h'. Variants of this method have proven very successful for object recognition in recent years.

3.2 Localization using Bag of Visual Words

As described in Section 2, we can use this classifier for localization by finding the subregion on which the decision function takes its maximal score. First, we rewrite the SVM's decision function f. Originally, this is

$$f(I') = \beta + \sum\nolimits_i \alpha_i \langle h', h^i \rangle$$

where $\langle ., . \rangle$ denotes the Euclidean scalar product in $\mathbb{R}^k$. α_i and β are the weight vector and bias that were learned during the training phase. Since $\langle h', h^i \rangle = \sum_k h'_k h^i_k$, f is a linear combination of contributions per histogram bin

$$= \beta + \sum\nolimits_{k=1}^{K} h'_k \underbrace{\sum\nolimits_i \alpha_i h^i_k}_{=:w_k}.$$

Algorithm 1: $O(n^2m)$ maximum sum algorithm

Require: $I \in \Re^{n \times m}$
Ensure: $(\hat{t}, \hat{b}, \hat{l}, \hat{r}) = \text{argmax}_{(t,b,l,r)} \sum_{t \le i \le b} \sum_{l \le j \le r} I_{i,j}$
  $M \Leftarrow IntImage(I)$
  $\hat{s} \Leftarrow -\infty$
  for $b = 1$ to n **do**
    for $t = 1$ to b **do**
      $s \Leftarrow -\infty$
      for $r = 1$ to m **do**
        if $s < 0$ **then**
          $l \Leftarrow r$
        end if
        $s \Leftarrow M_{b,r} - M_{b,l-1} - M_{t-1,r} + M_{t-1,l-1}$
        if $s > \hat{s}$ **then**
          $(\hat{t}, \hat{b}, \hat{l}, \hat{r}, \hat{s}) \Leftarrow (t, b, l, r, s)$
        end if
      end for
    end for
  end for

Using that h'_k is just the number of feature points in I' for which k was the closest cluster, we obtain the well-known fact that the score of a linear SVM can be written as a sum over per-point contributions:

$$f(I') = \beta + \sum_{j=1}^{n'} w_{c'_j}. \tag{1}$$

Here c'_j is the cluster label belonging to x'_j and n' is the number of feature points in I'. From this form, we see that f can be evaluated over subimages of I' by summing only over the feature points that fall into the subregion.

3.3 Conversion to a Matrix Problem

We can solve the problem of finding the rectangle $R \subset I'$ with maximal $f(R)$ by converting it into the classical problem of finding the subrectangle with maximal sum of entries in a matrix. For this, we build a sparse weight matrix M, where $m_{u,v}$ has the value $w_{c'_j}$ if x'_j is the u-th feature point when sorting the points in order of their x-coordinates, and the v-th feature point in order of their y-coordinates. The remaining entries of M are set to zero. Another way to think about this is to lay a grid with columns and rows of variable size over the image such that no two feature points lie in the same column or row of the grid. Treating each grid cell as a matrix element, we obtain the same representation as above. The sum over a subrectangle in this matrix corresponds to evaluating the sum in Equation (1) over the corresponding rectangle in image space.

4 The Maximum Sum Subrectangle Problem

4.1 Combinatoric Optimization

The maximum sum subrectangle problem is well studied, and we describe here a variant of the classical solution using $O(nm)$ space and $O(n^2m)$ time [27]. The initial step is to construct an integral image transform of the array. It can be computed in $O(nm)$ time and space, and allows us to evaluate the score within each subrectangle in constant time [12]. We could then iterate over all possible subrectangles in $O(n^2m^2)$ time, but it is more efficient to make use of the fact that the analogous 1D problem can be solved in linear time. We iterate over all possible pairs of the starting and ending indices for the rows of the matrix, and perform a 1D search for the starting and ending indices for the columns (Algorithm 1). In the case that the matrix is sparse, as in Section 3, we can reduce memory usage to $O(n)$ by implicitly computing the integral transform only for the portion of the image currently being explored. Sub-cubic algorithms rely on the basic form of this approach and then apply divide and conquer strategies [17, 18].

4.2 Branch-and-Bound Optimization

An alternative solution algorithm relies on branch-and-bound to reduce the number of candidate regions that have to be checked. The method we use is derived from Breuel's *Recognition by Adaptive Subdivision of Transformation Space (RAST)* algorithm [28]. The underlying idea is to evaluate many rectangles at the same time by bounding the maximal score that can be obtained on any of them. For this, we extend the rectangle representation to rectangle sets: instead of storing one index for each the left, right, top and bottom coordinate, we store intervals for these quantities. A set of four intervals represents the set of all rectangles that can be constructed by using elements from the intervals as corresponding rectangle coordinate. A sketch of this is given in Figure 1.

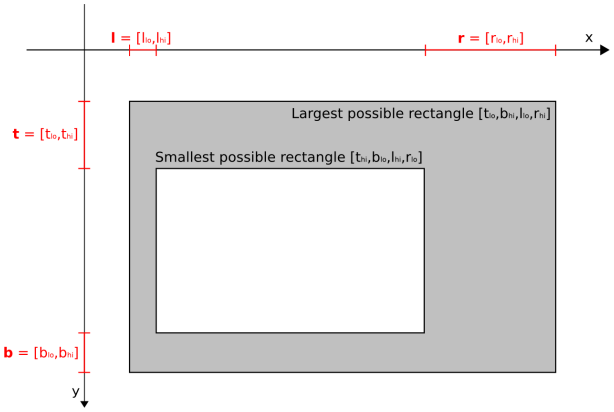


Figure 1: Representation of rectangle sets by four integer intervals.

Algorithm 2: Quality Function Optimization using RAST

Require: quality function f
Require: upper bound $\hat{f}$: $\hat{f}(\mathcal{R}) \geq \max_{(t,b,l,r)\in\mathcal{R}} f(t,b,l,r)$
Ensure: $(\hat{t},\hat{b},\hat{l},\hat{r}) = \text{argmax}_{(t,b,l,r)}\, f(t,b,l,r)$

Initialize P as empty priority queue
$(\mathbf{t},\mathbf{b},\mathbf{l},\mathbf{r}) \Leftarrow [1,n]\times[1,n]\times[1,m]\times[1,m]$
repeat
  split: $(\mathbf{t},\mathbf{b},\mathbf{l},\mathbf{r}) \Rightarrow (\mathbf{t}_1,\mathbf{b}_1,\mathbf{l}_1,\mathbf{r}_1)\,\dot{\cup}\,(\mathbf{t}_2,\mathbf{b}_2,\mathbf{l}_2,\mathbf{r}_2)$
  $P.enqueue(\,(\mathbf{t}_1,\mathbf{b}_1,\mathbf{l}_1,\mathbf{r}_1),\ \hat{f}(\mathbf{t}_1,\mathbf{b}_1,\mathbf{l}_1,\mathbf{r}_1)\,)$
  $P.enqueue(\,(\mathbf{t}_2,\mathbf{b}_2,\mathbf{l}_2,\mathbf{r}_2),\ \hat{f}(\mathbf{t}_2,\mathbf{b}_2,\mathbf{l}_2,\mathbf{r}_2)\,)$
  $(\mathbf{t},\mathbf{b},\mathbf{l},\mathbf{r}) \Leftarrow P.dequeue$
until $(\mathbf{t},\mathbf{b},\mathbf{l},\mathbf{r})$ consists of only a single rectangle $(\hat{t},\hat{b},\hat{l},\hat{r}) \Leftarrow (\mathbf{t},\mathbf{b},\mathbf{l},\mathbf{r})$

The RAST algorithm itself is a classical best-first branch-and-bound search as specified in Algorithm 2: in each iteration, the currently most promising rectangle set is extracted from a priority queue. It is split into two disjoint sets which are enqueued again, using upper bounds of their score as priority values. RAST is known to find a global maximum of a quality function f, if it has access to a function $\hat{f}$ that provides an upper bound for the maximum of f on rectangle sets and that converges to f when the set shrinks to a single rectangle. This holds not only for summing over subrectangles, but for a wide range of quality functions, see Section 6.

To construct a bounding function for our problem we use the monotony property of summation: for a given parameter set $\mathcal{R} = (\mathbf{t},\mathbf{b},\mathbf{l},\mathbf{r})$, we extract the largest possible rectangle $R_{max} = (t_{lo}, b_{hi}, l_{lo}, r_{hi})$ and the smallest possible rectangle $R_{min} = (t_{hi}, b_{lo}, l_{hi}, r_{lo})$. We then set

$$\hat{f}(\mathcal{R}) := f^{+}(R_{max}) + f^{-}(R_{min}) \tag{2}$$

Step	Method	Parameters
Preprocessing	RGB histogram normalization, gray-scale conversion	3% black/white quantiles
Feature Extraction	SURF keypoints and descriptors [19]	*threshold 100*, no rotational invariance
Codebook Generation	k-means clustering, concatenation of 10 per-class codebooks	random subset of 0.1% feature vectors $k = 300$
Image Representation	cluster histogram	3000 bins
Classifier Training	linear SVM	one-vs-rest, $C = 1000$
Localization	maximum sum subimage (1 per image)	pixel grid, no size penalty

Table 1: Classifier Details.

where f^+ and f^- are the sums over all positive or negative matrix entries in a rectangle. f^+ and f^- are monotonous functions with respect to the inclusion relation between rectangles, i. e. $f^+(R) \leq f^+(R^{max})$ and $f^-(R) \leq f^-(R^{min})$ for any $R \in \mathcal{R}$. Therefore,

$$\begin{aligned} f(R) &= f^+(R) + f^-(R) \\ &\leq f^+(R_{max}) + f^-(R_{min}) = \hat{f}(\mathcal{R}). \end{aligned}$$

showing that $\hat{f}$ is a bound for f from above. If $\mathcal{R}$ shrinks to a single rectangle R, we have $\hat{f}(\mathcal{R}) = f(R)$ because $R_{max} = R_{min} = R$. Therefore, $\hat{f}$ has both required properties for applying RAST. Using two pre-computed integral images, evaluating f^+ and f^- are $O(1)$ operations. Thus, $\hat{f}$ can be evaluated in constant time as well.

5 Experiments

So far we have shown how to convert classifiers into localizers in their functional form. We will show next that this is not only a very simple adaption, but the resulting method can also achieve very competitive results. For this, we implemented a setup that relies on a sparse bag of visual words representation, similar to the example of Section 3. The details for the underlying classifier are given in Table 1. As basis for our experiments we used the dataset of the PASCAL VOC Challenge 2006 [1], which consists of 5,304 images with 9,507 bounding box annotations in 10 object classes. The dataset is split into *train*, *val*, and *test* portions. All algorithm development was done on the *train* and *val* subsets, while *test* was reserved for the final evaluation.

The localization evaluation of the VOC challenge requires combined localization and ranking: the task is to return all bounding boxes of object instances from the *test* part of the dataset together with a confidence score. A box is said to be correctly localized if the ratio of the areas of the intersection and union between the prediction and ground truth is greater than 0.5. Then, for different *recall* levels, an *average precision (AP)* score is calculated.

Since the *test* dataset consists of images of all 10 object classes, it is strongly biased towards images that do not contain the current object class at all. Therefore, we first used a *bovw* classifier trained on the image level to order the images by decreasing SVM score. These scores express the confidence that an object of the class is present at all. We then performed localization by MSL using a different *bovw* classifier that was trained on *inside* versus *outside* of the ground truth bounding boxes.

For final results, we retrained the system using the *train* and *val* data combined and applied the resulting classifier to the previously untouched *test* part of the dataset. The scores achieved are reported in Table 2. The described procedure corresponds to the VOC2006 competition guidelines and allows us to compare our *average precision* scores with those of the competetition participants. As additional information, Table 3 shows the *total recall* and *total precision* of the localization system when detecting one image per class. The are calculated only for images that contain an object of the category and can be measured without ranking the detections by their confidence score.

All experiments were run on an ordinary PC with 2.4 GHz Intel Core Duo CPU under the Linux OS. The average runtime was $43ms$ per image per class for localization using branch-and-bound (not including feature extraction).

5.1 Localization Results

The performance on data VOC 2006 dataset is measured using precision–recall curves that are be combined into *average precision (AP)* scores, as reported in Table 2. For comparison, we also repeat the previous results from the VOC2006 challenge and related publications. In *Competition 3*, only VOC provided images were allowed for training. In *Competition 4*, arbitrary training material was allowed. We include results from both, since apart from

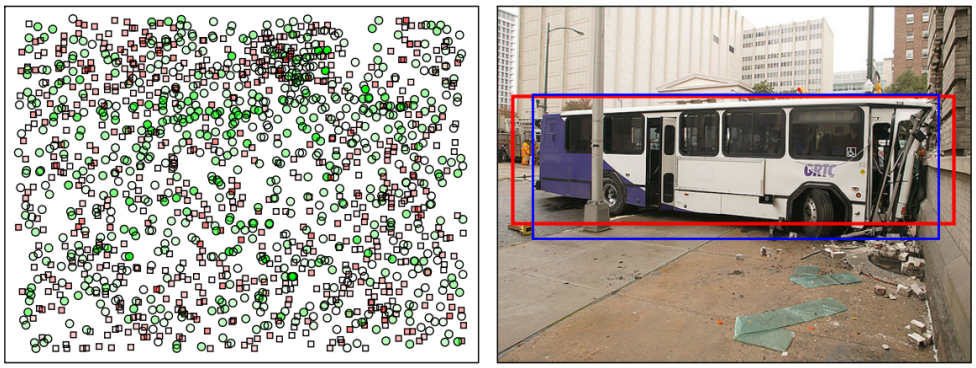


Figure 2: Successful *bus* localization. Left: Distribution of feature with visualization of scores. Green circles indicates positive, red boxes negative scores. Grey points have scores close to 0. Right: Localization results (red) and ground truth (blue).

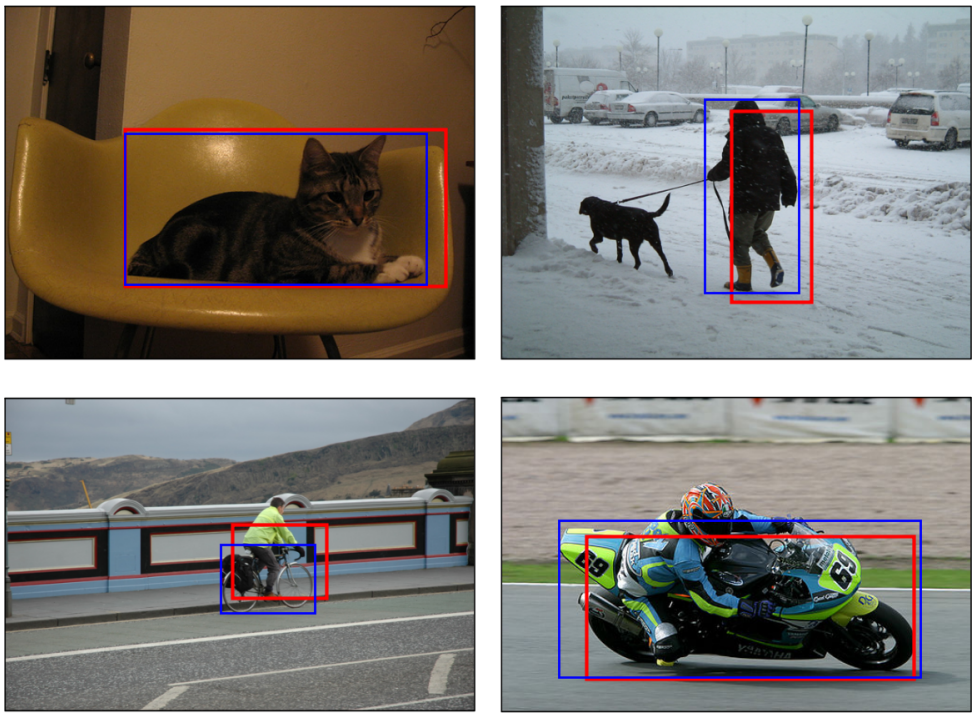


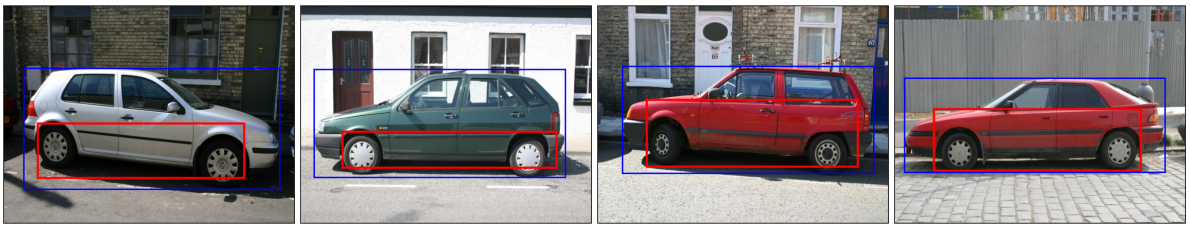
Figure 3: Further examples of successful object localization. Maximal subwindow found (red) and ground truth bounding box (blue).

the training material, the setting was identical, and the results are comparable. Our own method does not make use of additional training data and would therefore have fallen under *Competition 3*. In Table 3, we present results on the *total recall* and *total precision*. The *total recall* is the percentage of ground truth bounding boxes that our system correctly identified when returning one detection per image on the *test* data set. The criterion for a positive identification is an box-overlap of at least 50%, as specified in the VOC2006 rules. The *total precision* score is the percentage of detected boxes which correspond to actual ground truth object when MSL is applied only to image which do contain at least one object of the class to be located. This number therefore is closest to an actual measure of pure localization performance, since it measures the probability for a detected box to actually coincide with the location of an object.

To get a visual impression of the result achieved, Figures 2-4 show example images along with detections and ground truth. Using the criterion of box overlap, objects in Figures 2 and 3 were correctly identified, whereas the images in Figure 4 contain one or more objects that were counted as misclassifications.

(a) Object are missed if too few characteristic features points are detected on them. This can be due to insufficient texture (left), occlusion of important parts (center) or too small object size (right).

(b) Even with sufficiently many object features, localization errors occur: several objects are present, but we return only one per image (left), close or overlapping objects cause a single too big bounding box to be returned (center), small extended parts like legs can be missed (right).



(c) Cars have many near detections, but the overall precision score is poor. This is because the classifier concentrated its weights in the wheel region, causing the overlap with ground truth boxes to lie below 50%.

Figure 4: Typical reasons for localization mistakes: red boxes show the detection by MSL, blue boxes indicate ground truth data. Possible extensions to overcome these problems are discussed in Section 5.3

method	bicycle	bus	car	cat	cow	dog	horse	motorbike	person	sheep
Cambridge	0.249	0.138	0.254	0.151	0.149	0.118	0.091	0.178	0.030	0.131
ENSMP	-	-	0.398	-	0.159	-	-	-	-	-
INRIA-Douze	0.414	0.117	0.444	-	0.212	-	-	0.390	0.164	0.251
INRIA-Laptev	0.440	-	-	-	0.224	-	0.140	0.318	0.114	-
TUD	-	-	-	-	-	-	-	0.153	0.074	-
TKK	0.303	0.169	0.222	0.160	0.252	0.113	0.137	0.265	0.039	0.227
KUL *(Comp 4)*	-	-	-	-	-	-	-	0.229	-	-
MIT-Fergus *(Comp 4)*	-	-	0.160	-	-	-	-	0.159	-	-
MIT-Torralba *(Comp 4)*	-	-	0.217	-	-	-	-	-	-	-
Oxford-Chum [15]	0.493	0.249	0.412	-	0.212	-	-	0.371	-	0.195
Cornell-Crandall [29]	0.498	0.185	0.458	-	-	-	-	0.388	-	-
bag of visual words + MSL	0.380	0.178	0.080	0.165	0.161	0.109	0.091	0.260	0.005	0.117

Table 2: Combined localization and detection results on VOC-2006 dataset, see [1] for a description of the methods. Reported values are *average precision (AP)* scores as specified in the competition specifications. Methods marked with *"Comp 4"* made use of additional training images not provided by PASCAL. The other methods, including MSL, relied only on the material provided within the challenge for training. Oxford-Chum and Cornell-Crandall did not participate in the challenge but published results later using the same setup.

	bicycle	bus	car	cat	cow	dog	horse	motorbike	person	sheep
total recall	*0.475*	*0.356*	*0.156*	*0.499*	*0.337*	*0.426*	*0.330*	*0.555*	*0.079*	*0.180*
total precision	*0.599*	*0.508*	*0.275*	*0.566*	*0.619*	*0.494*	*0.463*	*0.726*	*0.161*	*0.425*

Table 3: Precision and Recall for MSL on images known to contain objects of the category. For each image, the system identified the box of highest classification score. A detected box counts as correctly identified if it's overlap with a ground truth box is at least 50%. *Total recall* is the percentage of objects that were found by MSL in this way. *Total precision* is the percentage of detections that correctly located an object. Since images can contain more than one object, the total recall is lower than the total precision.

5.2 Discussion of Results

The table shows that MSL based on *bovw* and a linear SVM achieves better results than all VOC2006 participants on two classes: *bus* and *cat*. Only after the contest, improved results for *bus* were published. For four further classes, *bicycle*, *cow*, *horse*, and *motorbike*, our results are comparable to the ones reported by the VOC-2006 participants. For the remaining classes *car*, *dog*, *person* and *sheep*, our results are worse than what was reported previously.

Taking into account that our method was not designed specifically for the VOC challenge or tuned to the diverse object classes, we believe that these are very respectable results. The technique relies entirely off-the-shelf components, in a way that can easily be reproduced. Also, with average run-times of less than $50ms$ per image, our method is certainly amongst the fastest for localizing arbitrary object classes in images.

Further insight is gained by comparing the achieved results with the *total recall* row. The *total recall* score is approximately the average precision score we would achieve with a ranking of the predicted bounding box that consistently scored correct predictions above erroneous ones. Since the recall is always clearly larger than the AP score achieved, we are confident that using a better method to rank the images would already improve the *AP* scores.

The system tested for on the VOC2006 data returns at most one object per class per image. Since there are many images with several instances of the same object class, we cannot expect to achieve 100% recall with the current setup. The row *total precision* gives the results how many of the boxes that were identified corresponded to actual objects in the images. The difference between these numbers and the *total recall* row shows how much gain we can expect by adding support for detecting multiple objects per image. Except for the *person* and *car* category, the precision scores achieved are all very promising, ranging from 42% to 72%. For the car dataset, the lower performance can be explained by a constant underestimation of the car region as is visible in Figure 4(c). We will comment on this in the following section. For the *person* category, none of the VOC2006 participants was able to achieve promising results. We believe that the intra-class variance of the dataset is too large in this case for a general object classification system. A system specialized in the detection of faces and/or human shapes might be more suitable here.

5.3 Problems and Solutions

Of course, our method does not achieve perfect localization recall. Examples of typical images where our setup does not identify (all) objects in an image correctly are shown in Figure 4. The purpose of this section is to explain the problems and indicate possible solutions.

The first class of problems are the result of insufficient or misleading features in the bag-of-visual-words representation of the image. Examples are given in Figure 4(a). In the leftmost image, the cat is not detected because its body is mainly unstructured an no interest points are detected onto it. To avoid this, one can use a feature representation that does not only rely on salient points, but also sample feature points from a regular grid or random locations in the image. This also allows to increase the total number of distinguishable patches that one can extract from an image. This is a desired effect, because it has been shown that the performance of object classification and localization methods can be improved by increasing the number of patches per image [30].

The image in the middle of Figure 4(a) shows a similar problem: the image background is strongly textured, which results in many more features being detected in the background than on the motorbike. Since also important parts of the motorbike are occluded, the random fluctuations of scores on background features dominated the positive scores on the motorbike, and the system false reports the whole image as an object. Again, using dense features can partly compensate for this, because the number of background features becomes more stable. Alternatively, a better base classifier might be needed, that is more robust to occlusions and diminishes the effect of background clutter.

The rightmost image of Figure 4(a) contains several sheep which are so small that even with a regular grid, only very few feature points will fall onto them. To nevertheless detect them, we currently investigate the use of information that can be extracted more reliably from very small image regions, e.g. color and texture. The result is a combination of different feature sets that can be combined linearly using Multiple-Kernel-Learning [31] and thereby integrated transparently into MSL.

Another class of problems occurs due to the very simple assumptions we used to introduce MSL. The left image of Figure 4(b) contains three cars (one of which is difficult to spot even for a human observer). Since our system returns only one object per image, the other two are missed, no matter how good our classifier is. An obvious solution is to iteratively allow the detection of several images. After the best rectangle has been found, the corresponding feature points are removed and MSL is started again. Each search returns its own score which can be used to determine how many objects were present at all. However, this method will not solve the problem of overlapping or very close objects, as is visible in the center image of Figure 4(b). MSL returns a regions that contains more than one object, leading to two or more missed localizations in the evaluation. To avoid this, we propose a geometric regularizer in the search. Trained on the shape that object bounding boxes are likely have (given by aspect ratio and area), it penalizes unlikely shapes in the image, e.g. very elongated structures, as they occur when many object overlap horizontally, see Section 6.2.

For some specific classes, inspecting the localization results shows other, object specific problem: e.g. the bounding boxes found for *cars* are consistently to small, including the wheels, but not the body of the car. This resulted in many erroneous matches that were very close to true ones (Figure 4(c)). A similar effect occurs for horses, where the body is identified much more reliably than the legs (Figure 4(b) right). The reason for this is the discriminative nature of our SVM-based base classifier: for cars, the wheels are the most discriminative part. The base classifier learns this by putting a high weight on features that resemble. When searching for the rectangle of maximum score, the wheel region is found, but the rest of the car might not, because it contains only features of very small positive, or even of negative weight.

As a possible fix, we are currently examining a post-processing step to the classification that performs a regression of the true bounding box based on the box of maximum score. This allows e.g. to learn that the car extends spatially above the wheel region, but not below it. Note that this is only necessary for discriminative base classifiers. A generative model would put weight onto the body of the car, even if the wheel region is more discriminative.

6 Extensions to other representations and classifiers

The object localization setup in Section 3 relied on a bag of visual words representation with a linear SVM classifier. However, the underlying principle of optimizing a classification function over subwindows can also be applied to many other representations and classifier systems. In this section, we give several further examples.

6.1 General Sparse Image Representations

From Equation (1) in Section 3 we see that neither the feature descriptor nor the classifier chosen matter as long as we can rewrite the decision function as a linear combination of individual contributions for each feature point. Another example of this would be the logarithm of the likelihood ratio in a naïve Bayes classifier. Since we are only searching for the maximum of the quality function, the method is also applicable for any monotonous transformation of linear feature combinations e. g. logistic regression.

6.2 Geometric Regularization

In addition to the data-dependent score of each subwindow, we can add terms to the objective function that depend only on the shape or position of the bounding box. This corresponds to a geometric regularization term as proposed in Section 5.3 to disallow unreasonable box shapes, even if their classifier score is high. Typically, such a regularizer could penalize the deviation of the box aspect ratio or size from a class specific mean. Note that we can see this still as a strict application of the MSL principle, where the underlying classifier now contains a data independent prior term about image shapes.

6.3 Dense probabilistic representations (Random Fields)

In a probabilistic setting, images are often represented not by sparse interest points, but values and labels are attached densely to each pixel location. Maximal Subwindow Search is applicable to this cases e.g. when there is statistical independence between the class conditional densities of different pixel locations when conditioned on their class labels. This assumption is common in probabilistic models like in Markov-Random-Fields (MRFs) or Conditional-Random-Fields (CRFs) [8] to make inference tractable at all. MSL can then find the maximum-a-posteriori solution from the space of all image labelings where the object regions is rectangular. For this, one chooses the the logarithm of the posterior probability as the quality function to be optimized, and obtain the sum of log-likelihood ratio of the class conditional densities to be maximized over all subwindows. An additional additive term derives from the prior and plays a similar role as the geometric regularization term introduced in the previous section.

6.4 Itemsets, Graphs and Binary Histograms

While the previous methods relied on certain locality assumptions (independence between feature points in *bovw* and conditional independence between pixels in the *MRF*), the method of subimage optimization can also be applied for representations that capture global properties like co-occurance or geometric arrangements [32, 33, 34].

Examples for this are weighted itemset and graph boosting [33]: a boosting classifier is learned based on weak classifiers that test for the co-occurance of different feature labels in the image. This makes the value of including a point into the region of interest depend on which other features are contained in it. Applying MSL, we cannot make use of integral images anymore, and the evaluation of the quality function is no longer a constant-time operation. Instead, each decision stump is evaluated in every step. The computational effort for this is kept moderate by keeping track of which feature combinations can contribute using match lists that are attached to search states, similar to [35]. Strongly related is the case where the underlying image representation are binarized feature histograms, because we can model the bins of the histograms as a itemsets contains only a single element.

6.5 Spatial Pyramid Matching Kernel

A very popular generalization of the the bag of words model is to include multi-level pyramid histograms based on the spatial distribution of points in the 2D image plane [36], called spatial pyramid matching kernel. Using branch-and-bound, we can maximize the pyramid score over subregions and perform MSL. For a single rectangle I, the decision function has the form

$$f(I) = \beta + \sum_{l=0}^{L} \beta_l \sum_{\substack{i=1,\dots l \\ j=1,\dots,l}} \sum_{k=1}^{N} \alpha_k \langle h'_{l,(i,j)}, h^k_{l,(i,j)} \rangle$$

where $h_{l,(i,j)}$ denotes the histogram of feature points within the (i,j)-th quadrant of a regular $l \times l$ split of the rectangle and β_l is a per-level weighting constant. Originally, it was chosen as $\beta_l = 2^{l-L}$, but recent results show that it is better to treat it as a free parameter during the learning process [37]

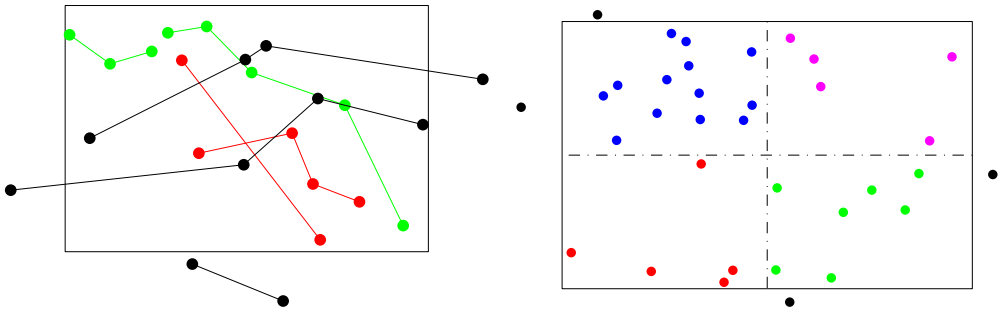


Figure 5: Left: Itemset scoring. Green itemsets contribute positively to the score, red ones negatively. Black itemsets do not contribute at all, because they are not fully contained in the subwindow. Right: Pyramid scoring. At level 0 all colorful points contribute a score depending on their label, but independent of their location. At level 1, a point's contribution also depends on which quadrant it lies in. Black points never contribute to the score.

As before, we obtain point-wise scores, now not only one value per point, but one per $(l, (i, j))$ multi-index, with separate sums that have to be calculated for each sub-part of the rectangle (Figure 5 right). For each branch-and-bound iteration, we compute upper bounds on each subwindow of a given pyramid level using $2 \times l \times l$ separate integral images per level. The upper bound of the set of candidates is the sum of the upper bounds of the subwindows.

6.6 Non Rectangular Shapes

In the area of object detection, axes-parallel boxes are the most common parametric shapes. While it would be possible to introduce additional parameter such as a rotation angle or a shear, it is unclear if this would improve the localization performance or rather increase the risk of overfitting.

In other areas of image understanding e.g. in biological applications or industrial computer vision, other parametric shapes than rectangles are sometimes preferred. The branch-and-bound optimization can be extended to these cases, as long as the number of parameters is fixed and not too large. This allows e.g. to efficiently search for circles or ellipses.

Additionally, non-parametric method for object localization have become more popular, e.g. exemplar based shape models. These could be integrated into MSL e.g. by binary masks overlaying the bounding box, and it will be interesting to see if the resulting algorithms can be made as efficient as the purely rectangle-based ones.

7 Conclusion

In this report we showed how existing methods for object classification can be turned into methods for object localization. The underlying idea of applying full image classifiers to subimages is so simple that it is amazing that it has not been studied systematically before.

We showed that despite being simple, the method achieves very competitive results for some object classes of the VOC2006 challenge, which measures combined detection and localization accuracy. Even for the classes where the method did not perform well, the localization accuracy itself is generally good when applied to only images in which an object is actually present. Using a better underlying classifier would even improve on this, and we believe that many existing object classification methods can be turned into successful methods for localization that way. It is our hope that many researchers with expertise in image classification will adopt the maximal subimage approach to add localization to their setup.

We imagine that the specific system we describe, with its simple feature extraction and classification pipeline out of off-the-shelf components, can serve as a baseline method with which future localization algorithms can compete. Furthermore, a set of classification algorithms that vary in a controlled way can be applied and the resulting performance can give an indication of the assumptions that are useful for localization in the context of specific datasets and object categories.

Acknowledgements

We would like to thank Sebastian Nowozin for his implementation of itemset boosting. This work was funded in part by the EC project CLASS, IST 027978. The first author is supported by a Marie Curie fellowship under the EC project PerAct, EST 504321.

References

[1] Mark Everingham, Andrew Zisserman, Chris Williams, and Luc Van Gool. The Pascal Visual Object Classes Challenge 2006 Results. Technical report, PASCAL Network, 2006. 1, 5, 8

[2] Stella X. Yu, Ralph Gross, and Jianbo Shi. Concurrent Object Recognition and Segmentation by Graph Partitioning. *NIPS 15*, pages 1383–1390, 2003. 1

[3] B. Leibe, A. Leonardis, and B. Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. In *ECCV'04 Workshop on Statistical Learning in Computer Vision*, pages 17–32, 2004. 1

[4] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to Detect Objects in Images via a Sparse, Part-Based Representation. *PAMI*, 26(11):1475–1490, 2004. 1

[5] Guillaume Bouchard and Bill Triggs. Hierarchical Part-Based Visual Object Categorization. In *CVPR*, pages 710–715, 2005. 1

[6] R. Fergus, P. Perona, and A. Zisserman. Weakly Supervised Scale-Invariant Learning of Models for Visual Recognition. *IJCV*, 71(3):273–303, Mar 2007. 1

[7] Nicolas Loeff, Himanshu Arora, Alexander Sorokin, and David Forsyth. Efficient Unsupervised Learning for Localization and Detection in Object Categories. In *NIPS 18*, pages 811–818, 2006. 1

[8] Sanjiv Kumar and Martial Hebert. Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification. In *ICCV*, pages 1150–1157, 2003. 1, 10

[9] Bryan C. Russell, William T. Freeman, Alexei A. Efros, Josef Sivic, and Andrew Zisserman. Using Multiple Segmentations to Discover Objects and their Extent in Image Collections. In *CVPR*, pages 1605–1614, 2006. 1

[10] Marcin Marszałek and Cordelia Schmid. Spatial Weighting for Bag-of-Features. In *oVPR*, pages 2118–2125, 2006. 1

[11] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human Face Detection in Visual Scenes. In *NIPS 8*, pages 875–881, 1996. 1

[12] Paul A. Viola and Michael J. Jones. Robust Real-Time Face Detection. *IJCV*, 57(2):137–154, 2004. 1, 3

[13] Shawn Newsam, Sitaram Bhagavathy, and B. S. Manjunath. Object Localization using Texture Motifs and Markov Random Fields. In *ICIP*, pages 1049–1052, 2003. 1

[14] C. Bouveyron, Juho Kannala, Cordelia Schmid, and Stephane Girard. Object Localization by Subspace Clustering of Local Descriptors. In *ICVGIP*, pages 457–467, 2006. 2

[15] O. Chum and A. Zisserman. An Exemplar Model for Learning Object Classes. In *CVPR*, pages 1–8, 2007. 2, 8

[16] Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999. 2

[17] T. Takaoka. Efficient Algorithms for the Maximum Subarray Problem by Distance Matrix Multiplication. In *CATS02*, pages 189–198, 2002. 2, 3

[18] Hisao Tamaki and Takeshi Tokuyama. Algorithms for the maximum subarray problem based on matrix multiplication. In *SODA*, pages 446–452, 1998. 2, 3

[19] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. SURF: Speeded Up Robust Features. In *ECCV*, pages 404–417, 2006. 2, 5

[20] Gy. Dorko and C. Schmid. Selection of Scale-Invariant Parts for Object Class Recognition. *ICCV*, pages 634–640, 2003. 2

[21] Fei-Fei Li and Pietro Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *CVPR*, pages 524–531, 2005. 2

[22] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. Van Gool. Modeling Scenes with Local Descriptors and Latent Aspects. In *ICCV*, pages 883–890, 2005. 2

[23] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering Objects and their Localization in Images. In *ICCV*, pages 370–377, 2005. 2

[24] Jianguo Zhang, Marcin Marszalek, Svetlana Lazebnik, and Cordelia Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *IJCV*, 73(2):213–238, 2007. 2

[25] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004. 2

[26] B. Schölkopf and A. Smola. *Learning With Kernels*. MIT Press, 2002. 2

[27] J. Bentley. *Programming Pearls*. Addison-Wesley, 1986. 3

[28] Thomas M. Breuel. Fast Recognition using Adaptive Subdivisions of Transformation Space. In *CVPR*, pages 445–451, 1992. 4

[29] D. J. Crandall and D. P. Huttenlocher. Composite models of objects and scenes for category recognition. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages 1–8, 2007. 8

[30] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV*, volume 3954 of *Lecture Notes in Computer Science*, pages 490–503. Springer, 2006. 9

[31] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004. 9

[32] David J. Crandall and Daniel P. Huttenlocher. Weakly Supervised Learning of Part-Based Spatial Models for Visual Object Recognition. In *ECCV*, pages 16–29, 2006. 10

[33] Sebastian Nowozin, Koji Tsuda, Takeaki Uno, Taku Kudo, and Gökhan Bakır. Weighted Substructure Mining for Image Analysis. In *CVPR*, 2007. 10

[34] T. Quack, V. Ferrari, and Luc J. Van Gool. Video Mining with Frequent Itemset Configurations. In *CIVR*, pages 360–369, 2006. 10

[35] Thomas M. Breuel. Implementation Techniques for Geometric Branch-and-Bound Matching Methods. *CVIU*, 90(3):258–294, Jun 2003. 10

[36] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, pages 2169–2178, 2006. 10

[37] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the International Conference on Image and Video Retrieval*, 2007. 10