Efficient Subwindow Search: A Branch and Bound Framework for Object Localization

Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann

***Abstract*—Most successful object recognition systems rely on binary classification, deciding only if an object is present or not, but not providing information on the actual object location. To estimate the object's location one can take a sliding window approach, but this strongly increases the computational cost, because the classifier or similarity function has to be evaluated over a large set of candidate subwindows.**

In this paper, we propose a simple yet powerful branch and bound scheme that allows efficient maximization of a large class of quality functions over all possible subimages. It converges to a globally optimal solution typically in linear or even sublinear time, in constrast to the quadratic scaling of exhaustive or sliding window search. We show how our method is applicable to different object detection and image retrieval scenarios. The achieved speedup allows the use of classifiers for localization that formerly were considered too slow for this task, such as SVMs with a spatial pyramid kernel or nearest neighbor classifiers based on the χ^2-distance. We demonstrate state-of-the-art localization performance of the resulting systems on the UIUC Cars dataset, the PASCAL VOC 2006 dataset and in the PASCAL VOC 2007 competition.

***Index Terms*—Object Localization, Sliding Window, Global Optimization, Branch and Bound.**

I. Introduction

RECENT years have seen great progress in the area of object category recognition for natural images. Recognition rates beyond 95% are the rule rather than the exception on many datasets. However, in their basic form, many state-of-the-art methods only solve a binary classification problem. They can decide whether an object is present in an image or not, but not where exactly in the image the object is located.

Object localization is an important task for the automatic understanding of images as well, *e.g.* to separate an object from the background, or to analyze the spatial relations of different objects in an image to each other. To add this functionality to generic object categorization systems, *sliding window* approaches have established themselves as state-of-the-art. Most successful localization techniques at the recent PASCAL VOC challenges on object category localization relied on this technique. The sliding window principle treats localization as localized classification, applying a classifier function subsequently to subimages within an image and taking the maximum of the classification scores as indicators for the presence of an object in this region. However, already an image of as low resolution as 320×240 contains more than *one billion* rectangular subimages. In general, the number of subimages grows quadratically with the number of image pixels, which makes it computationally too expensive to evaluate the quality function exhaustively for all of these. Instead, one typically uses heuristics to speed up the search that introduce the risk of mispredicting the location of an object or even missing it.

A similar problem exists in the field of image retrieval: existing methods for *content-based image retrieval* (CBIR) rely on global properties of images, *e.g.* color distribtions, or global statistics of local features, *e.g.* bag-of-words representations. Such methods typically fail when it is only a subregion of the image is of interest, *e.g.* a certain object or symbol as part of a larger scene.

In this paper, we propose *Efficient Subwindow Search* (ESS), a method for object localization that does not suffer from these drawbacks. It relies on a branch and bound scheme to find the global optimum of a quality function over all possible subimages in the possible candidate image, thus returning the same object locations that an exhaustive sliding window approach would. At the same time it requires much fewer classifier evaluations than there are candidate regions in the images—often even less than there are pixels— and typically runs in linear time or faster. Branch and bound optimization has been used in computer vision for geometric matching objectives [1]–[6], but we rather use branch and bound to optimize more general object localization objectives, including those based on quantized local features.

This paper extends [7] with additional empirical results and an in-depth analysis of ESS's performance compared to sliding window approaches. Sections II–IV explain how ESS allows the efficient maximization of an image quality functions over all subregions of an image. This enables object localization by localized classification and region-based image retrieval for quality functions that previously were considered unusable in these applications because they were too slow or had too many local maxima in their classification scores. Consequently, one obtains improved localization performance, as we will demonstrate in Sections V–VII. In the next section, we give an overview of other approaches for object localization and their relation to ESS.

A. Sliding Window Object Localization

Many different definitions of object localization exist in the scientific literature. Typically, they differ in the form that the location of an object in the image is represented, *e.g.* by its center point, its contour, a bounding box, or by a pixel-wise

Manuscript received XXX; revised YYY.

Christoph H. Lampert is with the Max Planck Institute for Biological Cybernetics, Tübingen, Germany. Matthew B. Blaschko is with the Department of Engineering Science, University of Oxford, United Kingdom. Thomas Hofmann is with Google Inc., Zürich, Switzerland.

segmentation. In the following we will only study localization where the target is to determine a bounding box around the object. This is a reasonable compromise between the simplicity of the parametrization and its expressive power for subsequent tasks like scene understanding. An additional advantage is that it is much easier to provide ground truth annotation for bounding boxes than *e.g.* for pixel-wise segmentations.

In the field of object localization with bounding boxes, sliding window approaches have been the method of choice for many years [8]–[12]. They rely on evaluating a quality function, *e.g.* a classifier's decision function, over many rectangular subregions of the image and taking its maximum as the object's location. Because the number of rectangles in an $n \times m$ image is of the order n^2m^2, one cannot check all possible subregions exhaustively. Instead, several heuristics have been proposed to speed up the search. Typically, these consist of reducing the number of necessary function evaluations by searching only with rectangles of certain fixed sizes as candidates and only over a coarse grid of possible locations [8]–[11]. Additionally, local optimization methods can be applied instead of global ones, by first identifying promising regions in the image and then using discrete gradient ascent procedure to refine the detection [12].

The reduced search techniques sacrifice localization robustness to achieve acceptable speed. Their implicit assumption is that the quality function is smooth and slowly varying. This can lead to false estimations or even complete misses of the objects locations, in particular if the quality function's maximum takes the form of a sharp peak in the parameter space. Note, however, that such a sharply peaked maximum is exactly what one would hope for to achieve accurate and reliable object localization.

II. Efficient Subwindow Search (ESS)

This section introduces *efficient subwindow search* (ESS), a technique to efficiently predict the best location of an object in an image for a fixed (usually trained) quality function. We start by formalizing the setup of window-based object detection. For this, we assume a quality function

$$f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \tag{1}$$

where $\mathcal{X}$ is the space of all images and $\mathcal{Y}$ is the space of rectangular subregions. We interpret $f(x, y)$ as the quality of the prediction that an object of the target class is located at position y in x. Note that we do not impose any *a priori* smoothness assumption on f.

We first study the situation where x is a single fixed image. Since no confusion can arise in this case, we use the notation $f(y)$ for $f(x, y)$. To predict the *best location* of the object, we have to solve

$$y_{opt} = \operatorname*{argmax}_{y \in \mathcal{Y}} \quad f(y). \tag{2}$$

Because $\mathcal{Y}$ has of the order $\mathcal{O}(n^2m^2)$ elements for an $n \times m$ image, we cannot perform this maximization exhaustively, except for very small images. Search based object detection methods like *sliding window* approximate the solution to Equation (2) by searching only over a small subset of $\mathcal{Y}$, which can result in suboptimal performance. In the following, we show that *efficient subwindow search* (ESS), which relies on a *branch and bound* scheme, can find the exact maximum of Equation (2) in a very computationally efficient way.

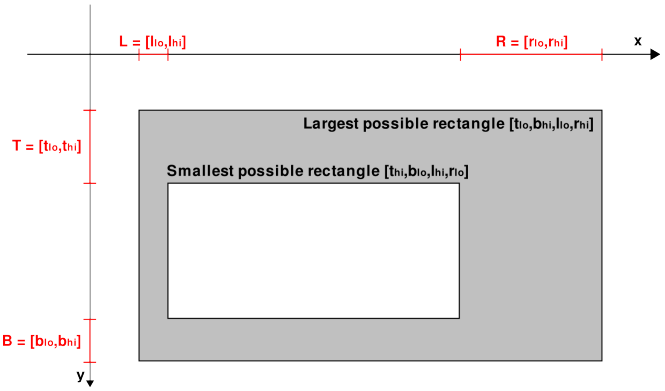


Fig. 1. Representation of rectangle sets by 4 integer intervals.

A. Branch and Bound Search

The insight behind ESS is an interpretation of Equation (2) not procedurally, *i.e.* not as a loop over all candidate regions, but mathematically as an optimization problem over a structured search space. This naturally leads to a targeted search towards the maximum instead of an exhaustive one. Since f might not be differentiable and can have many local maxima, we do not rely on local, *e.g.* gradient based techniques, but use a global *branch and bound* search [13].

The optimization works by hierarchically splitting the parameter space into disjoint subsets, while keeping bounds for the maximal quality for each of the subsets. Promising parts of the parameter space are explored first, and large parts of the parameter space do not have to be examined further if their upper bound indicates that they cannot contain the maximum.

In the case of ESS, the parameter space is the set of all possible rectangles, $\mathcal{Y}$, in an image, and subsets are formed by imposing restrictions on the values that the rectangle coordinates can take. We parameterize rectangles by their top, bottom, left and right coordinates (t, b, l, r), and we extend this parametrization to *sets* of rectangles by using intervals instead of single integers for each coordinate. This allows the efficient representation of sets of rectangles as tuples $[T, B, L, R]$, where $T = [t_{low}, t_{high}]$ *etc.*, see Figure 1 for an illustration. The full $n \times m$ image corresponds to the region $y \hat{=} [1, n, 1, m]$ in this representation, and $\mathcal{Y} \hat{=} [\,[1, n], [1, n], [1, m], [1, m]\,]$.

For each rectangle set, we calculate a bound for the highest score that the quality function f could take on any of the rectangles in the set. ESS terminates when it has identified a rectangle with a quality score that is at least as good as the upper bound of all remaining candidate regions. This criterion guarantees that a global maximum has been found.

ESS organizes the search over candidate sets in a *best-first* manner, always examining next the rectangle set that looks most promising in terms of its quality bound. The candidate set is split along its largest coordinate interval into halves, thus forming two smaller disjoint candidate sets (Figure 2). The

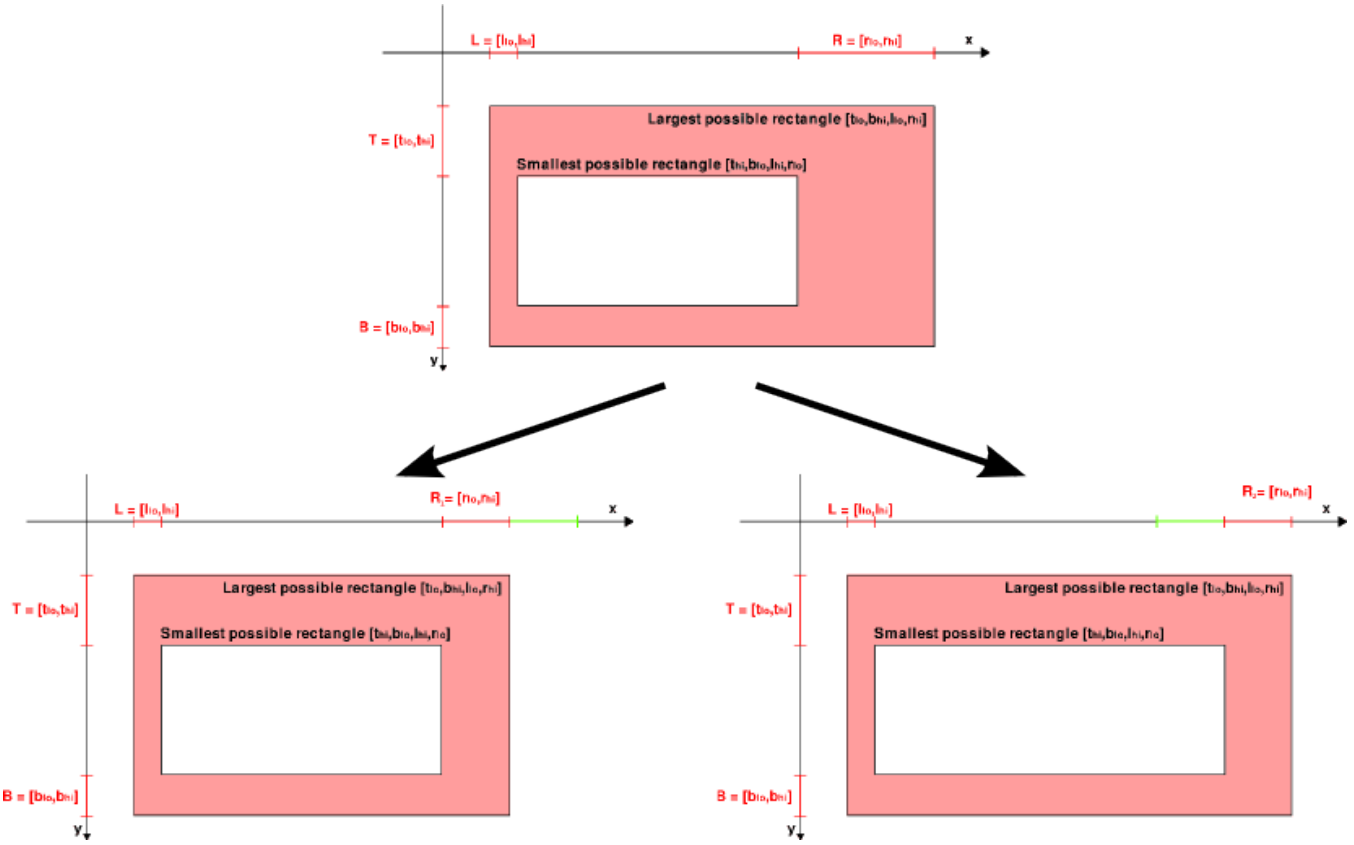


Fig. 2. Splitting rectangle sets is done by dividing one of the intervals in two. In this case, $[T, B, L, R] \rightarrow [T, B, L, R_1] \dot{\cup} [T, B, L, R_2]$, where $R_1 := [r_{lo}, \lfloor \frac{r_{lo}+r_{hi}}{2} \rfloor]$ and $R_2 := [\lfloor \frac{r_{lo}+r_{hi}}{2} \rfloor + 1, r_{hi}]$.

search is stopped when the most promising set contains only a single rectangle with the guarantee that this is the rectangle of globally maximal score. This form of branch and bound search has been shown to require the minimal possible number of function evaluations [14] in the setup chosen. Algorithm 1 gives pseudo-code for ESS using a priority queue to hold the search states.

III. Construction of Quality Bounding Functions

ESS is a completely generic optimization technique. It can be applied to any quality function f, for which we can construct a function that upper bounds the values of f over sets of rectangles $Y \subset \mathcal{Y}$. This bounding function $\hat{f}$ has to fulfill the following two properties:

$$i) \quad \hat{f}(Y) \geq \max_{y \in Y} f(y),$$

$$ii) \quad \hat{f}(Y) = f(y), \quad \text{if } y \text{ is the only element in } Y.$$

Condition $i)$ ensures that $\hat{f}$ acts as an upper bound on f, whereas condition $ii)$ guarantees the optimality of the solution to which the algorithm converges. In practice, $\hat{f}$ only has to be defined for rectangles sets Y that have a $[T, B, L, R]$ representation, as only these can occur during the algorithm.

Note that for any f there is a spectrum of possible bounding functions $\hat{f}$. On the one end, one could perform an exhaustive search to achieve exact equality in $i)$. On the other end, one could set $\hat{f}$ to a large constant on everything but single rectangles. A good bound $\hat{f}$ is located between these extremes: fast to evaluate but also tight enough to ensure fast convergence. In the following sections we show how such bounding functions $\hat{f}$ can be constructed for different choices of f.

Algorithm 1 Efficient Subwindow Search

Require: image x
Require: quality bounding function $\hat{f}$ (see Sect.III)
Ensure: $(t_{\text{opt}}, b_{\text{opt}}, l_{\text{opt}}, r_{\text{opt}}) = \text{argmax}_{y \in \mathcal{Y}} f(y)$
  initialize P as empty priority queue
  set $[T, B, L, R] = [1, n] \times [1, n] \times [1, m] \times [1, m]$
  repeat
    split $[T, B, L, R] \rightarrow [T_1, B_1, L_1, R_1] \dot{\cup} [T_2, B_2, L_2, R_2]$
    push $([T_1, B_1, L_1, R_1]; \hat{f}([T_1, B_1, L_1, R_1]))$ onto P
    push $([T_2, B_2, L_2, R_2]; \hat{f}([T_2, B_2, L_2, R_2]))$ onto P
    retrieve top state $[T, B, L, R]$ from P
  until $[T, B, L, R]$ consists of only one rectangle
  set $(t_{\text{opt}}, b_{\text{opt}}, l_{\text{opt}}, r_{\text{opt}}) = [T, B, L, R]$

A. Linear Classifiers

As demonstration of how to construct the necessary quality bounding function, we first study the case where the quality function f is the decision function of a support vector machine (SVM) with a linear kernel over a *bag-of-visual-words (bovw)*

histogram representation. Each image x is represented by a set of feature points d_j, $j = 1, \ldots, n$, where for each feature point we store its image coordinates and a bag-of-visual-words cluster id c_j. Given any rectangular region y in x, we can form $x|_y$, *i.e.* the image x cropped to the region y, which is again an image and some of the feature points lie inside it. For any such $x|_y$, we can form the K-bin histogram $h = h(x|_y)$, where each entry h_k counts how many feature points of the cluster id k occur in $x|_y$. Such *bovw*-histograms will be the underlying representations for all quality functions that we study in this section. Note that for simplicity, we use unnormalized histograms in this construction, except where indicated otherwise.

In its canonical form, the corresponding SVM decision function is $f(h) = \beta + \sum_i \alpha_i \langle h, h^i \rangle$, where $\langle ., . \rangle$ denotes the scalar product in $\mathbb{R}^K$. h^i are the *bovw*-histograms of the training examples and α_i and β are the constant weights and bias term that are learned during SVM training. Because of the linearity of the scalar product, we can rewrite this expression as a sum over per-point contribution with weights $w_m = \sum_i \alpha_i h^i_m$:

$$f(y) = \beta + \sum_{d_m \in x|_y} w_{c_m}. \tag{3}$$

where the sum runs over all feature points d_m that lie in the region y. Because we are only interested in the argmax of f over all $y \in \mathcal{Y}$ (Equation (2)), we can drop the constant bias term β.

It is now straight-forward to construct a function $\hat{f}$ that bounds f over sets of rectangles $Y \subseteq \mathcal{Y}$. First, we decompose $f = f^+ + f^-$, where f^+ contains only the positive summands of Equation (3) and f^- only the negative ones. For a set of regions Y, we denote by $y_\cup$ the union of all rectangles in Y and by $y_\cap$ their intersection. Then

$$\hat{f}(Y) \equiv f^+(y_\cup) + f^-(y_\cap) \tag{4}$$

will be a bound for f that fulfills the criteria $i)$ and $ii)$. Checking property $ii)$ is trivial, in this case, since $y_\cup = y_\cap = y$ if $Y = \{y\}$, and $f^+(y) + f^-(y) = f(y)$ by construction. To show $i)$, we observe that for any $y \in Y$ the feature points that lie in y are a subset the points in $y_\cup$ and a superset of the points in $y_\cap$. Since f^+ contains only positive summands, we have $f^+(y_\cup) \geq f^+(y)$, and analogously $f^-(y_\cap) \geq f^-(y)$ because f^- contains only negative summands. In combination, we obtain that

$$\hat{f}(Y) = f^+(y_\cup) + f^-(y_\cap) \geq f(y) \tag{5}$$

holds for any $y \in Y$ and therefore also for the element maximizing the right handside.

To make $\hat{f}$ a useful quality bounding function, we have to show that we can evaluate it efficiently for arbitrarily large $Y \in \mathcal{Y}$. If Y was an arbitrary set of rectangles, finding $y_\cup$ and $y_\cap$ could require iterating over all elements. However, rectangle sets in the ESS algorithm are always given in their parametrization $[T, B, L, R]$. This ensures that $y_\cup$ and $y_\cap$ will be rectangles again, which is important in order to efficiently represent them. It also allows us to determine $y_\cup$ and $y_\cap$ in constant time: writing $T = [t_{low}, t_{high}]$, *etc.*, one sees that $y_\cup = [t_{low}, b_{high}, l_{low}, r_{high}]$ and $y_\cap = [t_{high}, b_{low}, l_{high}, r_{low}]$. If the latter is not a legal representation of a rectangle, *i.e.* if $r_{low} < l_{high}$ or $b_{low} < t_{high}$, then $y_\cap$ is empty.

Using integral images [15] we can make the evaluations of f^+ and f^- constant time operations, thus making each evaluation of $\hat{f}$ an $\mathcal{O}(1)$ operation. The fact that the evaluation time of $\hat{f}$ is *independent of the number of rectangles* contained in Y is a crucial factor in why ESS is fast.

B. Spatial Pyramid Features

Raw bag-of-visual-words models, as used in the previous section, have no notion of *geometry*. They are therefore not the best choice for the detection of object classes which have characteristic geometric arrangements, *e.g.* cars or buildings. Spatial pyramid features have been developed to overcome this limitation. They divide every image into a grid of regions and represent each grid cell by a separate histogram. Typically, a pyramid of increasingly fine subdivisions is used, see [16] for the exact construction.

We consider an SVM classifier with linear kernel on top of such a hierarchical spatial pyramid histogram representation. The decision function f for a region y in an image x is calculated as

$$f(y) = \beta + \sum_{l=1}^{L} \sum_{\substack{i=1,\ldots l \\ j=1,\ldots,l}} \sum_{k=1}^{N} \alpha_k^{l,(i,j)} \langle h^y_{l,(i,j)}, h^k_{l,(i,j)} \rangle, \tag{6}$$

where $h^y_{l,(i,j)}$ is the histogram of all features of the image x that fall into the spatial grid cell with index (i, j) of an $l \times l$ spatial pyramid in the region y. $\alpha_k^{l,(i,j)}$ and β are the coefficients learned by the SVM when trained with training histograms $h^k_{l,(i,j)}$.

Using the linearity of the scalar products, we can again transform this into a sum of per-point contributions:

$$f(y) = \beta + \sum\nolimits_{m=1}^{n} \sum_{l=1}^{L} \sum_{\substack{i=1,\ldots l \\ j=1,\ldots,l}} w_{c_m}^{l,(i,j)}, \tag{7}$$

where $w_{c_m}^{l,(i,j)} = \sum_k \alpha_k^{l,(i,j)} h^k_{l,(i,j);c_m}$, if the feature point d_m has cluster label c_m and falls into the (i, j)-th cell of the l-th pyramid level of y. Otherwise, we set $w_{c_m}^{l,(i,j)} = 0$. As before, we can ignore the bias term β for the maximization over $y \in \mathcal{Y}$.

A comparison with Equation (3) shows that Equation (7) is a sum of *bovw* contributions, one for each level and cell index (l, i, j). We bound each of these as explained in the previous section: for a given rectangle set Y, we calculate box regions containing the intersection and union of all grid cells $y^{l,(i,j)}$ that can occur for any $y \in Y$. Calling these $y_\cup^{l,(i,j)}$ and $y_\cap^{l,(i,j)}$, we obtain an upper bound for a cell's contribution by adding all weights of the feature points with positive weights $w_c^{l,(i,j)}$ that fall into $y_\cup^{l,(i,j)}$ and the weight of all feature points with negative weights that fall into $y_\cap^{l,(i,j)}$. An upper bound $\hat{f}$ for f is obtained by summing the bounds for all levels

and cells. If we make use of two integral images per triplet (l, i, j), evaluating $\hat{f}(Y)$ becomes an $\mathcal{O}(1)$ operation. This shows that also for the spatial pyramid representation, efficient localization using ESS is possible.

C. Non-linear additive classifiers

Many window-based object detection systems can benefit from non-linear classifiers to achieve better performance. ESS is applicable to these as well, if a suitable bound is available. In this section, we show how to construct exemplary bounds for the *histogram intersection kernel* and χ^2*-distance*. The former is popular in the context of the *pyramid match kernel* [17]. The latter has been used *e.g.* for nearest-neighbor based classifiers [18], but by setting $k(h, h') = -\chi^2(h, h')$, it can also be used as the kernel of an SVM classifier.

1) (Generalized) histogram intersection kernel: The *generalized histogram intersection kernel* [19] is defined as

$$k_{GHI}(h, h') = \sum_{k=1}^{K} [\min(h_k, h'_k)]^\gamma. \tag{8}$$

where $\gamma > 0$ is a normalization parameter. For $\gamma = 1$ we obtain the ordinary *histogram intersection measure* [20], [21]. To use this kernel for ESS localization, we need to construct bounds for

$$f(y) = \sum_{j=1}^{n} \alpha_j \sum_{k=1}^{K} [\min(h_k^j, h_k^y)]^\gamma, \tag{9}$$

where h^j are fixed training histograms and h^y is the histogram of the cropped image $x|_y$, and y varies within a candidate set Y. As before, we have ignored the SVM's bias term.

Notice at first that the value of each histogram bin h_k^y for $y \in Y$ can be bounded from above and from below by the number of keypoints with corresponding cluster index that fall into $y_\cup$ and $y_\cap$ respectively. We denote these bounds by $\overline{h}_k^Y$ and $\underline{h}_k^Y$. Thus, we obtain

$$\min(h_k, \underline{h}_k^Y) \leq \min(h_k, h_k^y) \leq \min(h_k, \overline{h}_k^Y) \tag{10}$$

with equality in the situation that $Y = \{y\}$. This implies that for any $\gamma > 0$, we can now bound the summands in Equation (8) from above and from below by

$$[\min(h_k, \underline{h}_k^Y)]^\gamma \leq [\min(h_k, h_k^y)]^\gamma \leq [\min(h_k, \overline{h}_k^Y)]^\gamma. \tag{11}$$

Consequently,

$$\hat{f}(Y) = \sum_{\alpha_j > 0} \alpha_j [\min(h_k, \overline{h}_k^Y)]^\gamma + \sum_{\alpha_j < 0} \alpha_j [\min(h_k, \underline{h}_k^Y)]^\gamma \tag{12}$$

is a quality bound for Equation (9) that fulfills $i)$ and $ii)$.

2) χ^2-distance and kernel: The χ^2-distance between two histograms is calculated from the squared distance between the bins, reweighted in a data dependent way. In contrast to the kernels used previously, it is common to normalize the histograms before calculating their distance, giving them the properties of empirical probability distributions:

$$\chi^2(h, h') = \sum_{k=1}^{K} \frac{(p_k - p'_k)^2}{p_k + p'_k} \tag{13}$$

with $p_k \equiv \frac{1}{\sum_k h_k} h_k$ and $p'_k \equiv \frac{1}{\sum_k h'_k} h'_k$. To construct a bound over a set of boxes Y, we first use the same construction as for the intersection kernel to obtain unnormalized per-bin bounds $\overline{h}_k^Y$ and $\underline{h}_k^Y$. We can bound each normalized entry by $\underline{p}_k \leq p_k \leq \overline{p}_k$ by

$$\underline{p}_k^Y \equiv \frac{1}{\max\{1, \underline{h}_k^Y + \sum_{k' \neq k} \overline{h}_{k'}^Y\}} \underline{h}_k^Y, \tag{14}$$

$$\overline{p}_k^Y \equiv \frac{1}{\max\{1, \overline{h}_k^Y + \sum_{k' \neq k} \underline{h}_{k'}^Y\}} \overline{h}_k^Y. \tag{15}$$

Each component of the χ^2-distance is bounded from below by

$$\frac{(p_k - p_k^Y)^2}{p_k + p_k^Y} \geq \begin{cases} (p_k - \underline{p}_k^Y)^2 / (p_k + \underline{p}_k^Y) & \text{for } p_k < \underline{p}_k^Y, \\ 0 & \text{for } \underline{p}_k^Y \leq p_k \leq \overline{p}_k^Y, \\ (p_k - \overline{p}_k^Y)^2 / (p_k + \overline{p}_k^Y) & \text{for } p_k > \overline{p}_k^Y, \end{cases} \tag{16}$$

The negative sum of these expressions fulfills properties $i)$ and $ii)$ for a quality function $f(y) = -\chi^2(h, h^y)$. For use in a support vector machine, one forms an upper bound in analogue to (16) and combines both as has been done in Equation (12) for the histogram intersection kernel, splitting the linear combination into positive and negative contributions. Note that bounds based on such chained constructions as for χ^2-distance are generally looser than direct ones, and the branch and bound search typically requires more iterations to converge than for the linear kernels used previously. Althought both bounds in this section require more computation than in the linear cases, they can nevertheless be evaluated efficiently by using integral histograms [22]. However, this comes at the expense of highly increased memory usage, which can become prohibitive for very large *bovw* codebooks. A promising alternative way has been opened by Maji et al. [23], who derived an efficient evaluation of the quality function based on interchanging the order of summations in Equations (9). It can be presumed that a similar construction would be possible for the bound calculation as well.

D. Quality bounds by interval arithmetic

Another powerful approach to obtain quality bounding functions for nearly arbitrary quality functions is *interval arithmetic*, see *e.g.* [24], [25]. It allows computation with uncertain quantities, in our case the intervals used to represent rectangle sets. Breuel [26] applied this idea to a specific quality function for the detection of geometric objects in line drawings.

An advantage of interval arithmetic is the reduced human effort in constructing a bound and a reduced risk of error in implementing it. With existing class or template libraries, interval computations can be performed transparently, with the same routines that perform single evaluations of the quality function. On the downside, bounds that are derived automatically are usually less tight than those constructed manually, causing a slowdown of the branch and bound search.

IV. Extensions of ESS

Several extensions of the basic ESS search scheme are possible in order to provide additional functionality. To favor boxes with specific shape properties one can add a geometric penalization term to f. Typically, this could be a Gaussian that takes its maximum at a certain rectangle size or aspect ratio. Of course this term has to be taken in account when deriving the bound for f. An alternate approach is to modify the topology of a bounding region, e.g. as has been proposed for piecewise linear bounding regions [27].

Object localization tasks often require the detection of more than one object location in an image. For this, we can apply Algorithm 1 repeatedly: whenever an object is found, all feature points of the corresponding region are removed from the image and the search is restarted until the desired number of locations have been returned. Alternatively, one can simply continue the search after the best location has been identified, detecting the second best, third best region, *etc.*. However, this requires a non-maximum suppression step, similar to what sliding window approaches do to achieve multiple detections.

Finally, ESS can also be parallelized to make better use of multi-core CPUs, HPC clusters or even computation on the GPU. See *e.g.* [28] for a survey of techniques to parallelize branch and bound algorithms.

A. Simultaneous ESS for multiple images and classes

A common situation in realistic applications is that one does not have to evaluate only one quality function for one test image, but rather several quality functions, *e.g.* for multi-class classification, or one has to process many images, *e.g.* an image database. Formally, this situation can be written as

$$(y_{opt}, x_{opt}, \omega_{opt}) = \operatorname*{argmax}_{\substack{y \in \mathcal{Y},\, \omega \in \Omega \\ x \in \{x_1, \ldots, x_n\}}} f_\omega(x, y), \qquad (17)$$

where each f_ω is a quality function for a class ω from a set of classes Ω that are to be detected[1], and x ranges over all images in an image collection $\{x_1, \ldots, x_N\}$. Other index sets would, of course, also be possible.

A straight-forward application of ESS to this situation would be to search for the best region for each class in each image and choose the one with largest quality score. However, we can achieve a much more efficient search by interleaving the maximization over the best region within each image with the maximizations over all images and all classes into a single best-first search. For this, we add the start states of all images and all classes into the priority queue before starting the search.[2] While ESS runs, the candidate regions of all images and classes are simultaneously brought into an order according to how relevant they are to the query. Search states that do not contain promising candidate regions always stay at the bottom of the queue and might never be expanded.

In retrieval scenarios, one is interested not only in the single best result, but *e.g.* the top N images containing an object. This is possible by a continued search approach: whenever an optimal match has been found, we remove the states corresponding to the image found from the search queue and continue the search until N regions have been detected.

Note that the appealing idea of extending the full branch and bound search to also cover Ω and $\{x_1, \ldots, x_N\}$ is not possible in general. Since these domains do not have a geometric structure as $\mathcal{Y}$ does, it is unclear how to come up with non-trivial bounds for the quality function across different images and classes. Domain dependent solutions could, however, be possible, *e.g.* for video sequences where subsequent frames are strongly correlated.

V. Application I: Localization of arbitrary object categories

To demonstrate the performance of ESS in terms of speed and accuracy, we apply it to several realistic problem settings from the areas of object localization and image retrieval. We start by building a system that performs localization of arbitrary object categories. It uses an SVM classifier based on the *bag of visual words* image representation as quality function, as has been introduced in Section III. By choosing the *bovw* representation, all relative spatial information between feature points in an image is disregarded and the detection system becomes invariant to any changes in the object geometry, pose and viewpoint. Consequently, we obtain a localization system that in principle is robust enough to detect arbitrary objects. In particular, it is eligible for the detection of object classes that show a large amount of variance in their visual appearance as is the case, *e.g.*, for many natural objects and animals.

A. PASCAL VOC 2006 dataset

In a first set of experiments, we tested the *bovw* based localization on the *cat* and *dog* categories of the publicly available PASCAL VOC 2006 dataset[3], see Figure 3 for examples. The dataset contains 5,304 natural images, which are split into *training* and *validation* parts, on which all algorithm development is performed, and a *test* part that is reserved for the final evaluation. 1,503 images in the set show at least one cat or dog. Some of the images contain more than one object, and in total, there are 1,739 object intances.

To represent the images we extract SURF features [29] from keypoint locations and from a regular grid and quantize them using a 1000 entry codebook that was created by K-means clustering a random subset of 50,000 descriptors. As

[1]Note that query by *multiple* examples is a special case of this setting in which the different classes represent different query examples.

[2]If the number of images is so large that we exceed the cache, we may get counterintuitive run-time performance. In these cases it may be more efficient to use a parallel branch and bound strategy on multiple nodes [28].

[3]http://www.pascal-network.org/challenges/VOC/voc2006/

Fig. 3. Example images of `cat` (top) and `dog` (bottom) categories of PASCAL VOC 2006 dataset. Objects occur in different sizes and poses, and multiple object instances are possible within one image. Objects are also frequently occluded or truncated.

positive training examples for the SVM we use the ground truth bounding boxes that are provided with the dataset. As negative examples we sample 4,500 box regions from images with negative class label and from locations outside of the object region in positively labeled images. From this we trained a support vector machine with linear kernel, using the *validation* part of the dataset to select regularization parameter $C \in \{10^{-3}, \ldots, 10^{3}\}$.

1) ESS vs. Sliding Window Localization: As we have seen in the previous sections, ESS has the advantage over sliding window methods that it finds a global optimum of the quality function. Of course, this is only relevant if ESS is also fast enough to be used in practical application, and if finding the better maximum translates into better localization performance. Therefore, we first benchmark ESS against sliding window detectors in terms of these two properties.

In contrast to ESS, sliding window methods require the specification of several parameters, in particular the set of window sizes and aspect ratios to search over, and the step widths of the search grid. By choice of these parameters, sliding window methods can be made arbitrarily fast or slow. This is achieved by trading off that less or more candidate regions in the image are checked. For our experiments we implemented five representative combinations of parameters ($SW_1, \ldots, SW_5$, see Table I). They are similar to what is used in practical object localization systems [10], [15], [30], and, in particular, the values are chosen such that the resulting systems remain computationally tractable.

At first, we compare only the speed of ESS against the sliding window approaches. To be independent of the hardware and actual implementation of the quality function, we measure *speed* by the number of quality function evaluations performed. Evaluations of the quality bound (4) are counted as two evaluations of the quality function itself. The total number of evaluations varies with the image sizes and, in case of ESS, the image content. To obtain comparable scores, we use the scale-free *ratio* of function evaluations n^{ESS}/n^{SW_i} for each set of sliding window parameters, $i = 1, \ldots, 5$. Figure 4(a) shows the results of applying the *cat* and *dog* detection to all test images of the PASCAL VOC 2006 dataset. Blue bars in the histogram indicate that ESS required fewer evaluation than SW_i, and red bars indicate the opposite.

Next, we compare the value of the *quality* function found by both detection methods. It is clear by construction that ESS will return better quality scores, as it finds the global maximum of the quality function whereas sliding window methods most often will not. We therefore use the score $f(y^{ESS})$ found by ESS as reference value and compare how close the values $f(y^{SW_i})$ returned by the sliding windows approaches come to it. Again, we make the values scale independent by taking ratios $f(y^{SW_i})/f(y^{ESS})$. The results are plotted in Figure 4(b).

The main quantity that we are interested in when performing object localization is not the value of the quality function, but the quality of the box detections. Again we know that ESS finds the globally best region y^{ESS} in the sense of f, whereas the regions found by sliding windows y^{SW_i} might be suboptimal. To measure how far both detections differs, we calculate the *area overlap* between ESS's detection and the regions returned by the sliding window methods:

$$\text{overlap}(y, y') = \frac{\text{Area}(y \cap y')}{\text{Area}(y \cup y')} \tag{18}$$

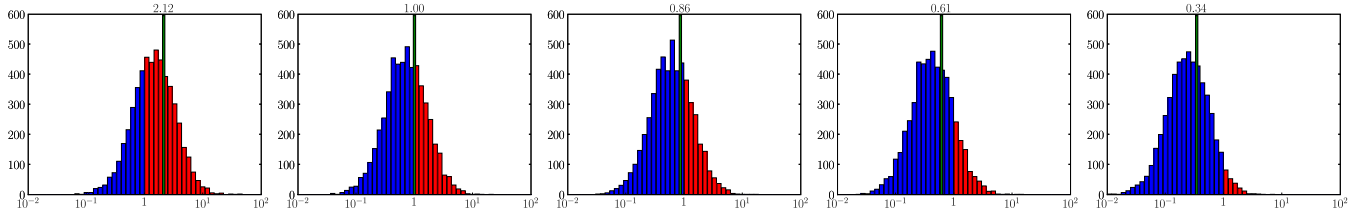
Figure 4(c) shows the distribution of overlap scores.

Any region-based method for object localization can achieve only as good results as the quality function used allows. To ensure that the differences in performance between evaluation procedures is also present for practical purposes, we compare the detected regions y^{ESS} and y^{SW_i} with the ground truth regions y^{gt} of the dataset. Figure 5(a) shows a scatter plot of the overlaps and Figure 5(b) shows the distribution of the differences in the overlap scores. In both plots, the cases where ESS achieves a higher overlap with the ground truth than the sliding window approach are drawn in blue, and the opposite cases in red.

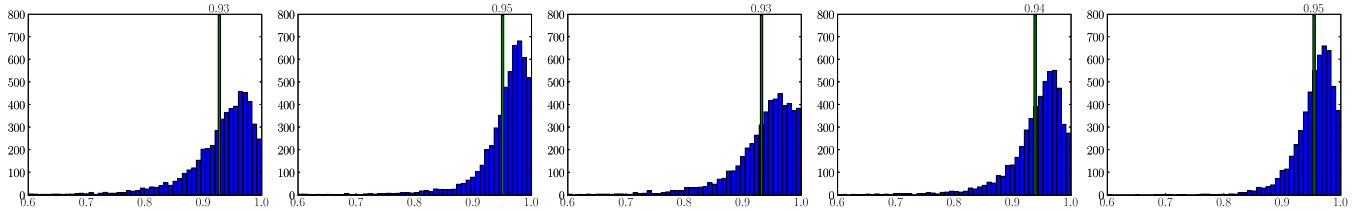
Overall, the Figures 4 and 5 allow us to draw several conclusions. First, since ESS is globally optimal whereas sliding window method are not, it is not surprising that ESS achieves better quality scores. Also, if one makes the sliding window search to search more and more boxes, the method naturally becomes slower and slower. However, one would expect that with more window evaluation, the sliding window quality scores and the overlap with the optimal rectangles would approach the ESS results. This does not seem to be the case. If Figure 4 shows such a trend at all, it would have to be very weak. This might be due to the fact that any feasible sliding window technique, even if it performs several times the number of evaluation that ESS requires, can still only sample a very small fraction of the full search space of all image regions, and is therefore far from convergence to ESS's results.

Figure 5 shows a similar picture. Sliding window methods with fewer evaluations do not automatically achieve worse detection results than those with more window evaluations. All of the sliding window methods return boxes that on average have significantly lower overlap with the ground truth than the regions found by ESS. However, the figure also shows that learning a good quality function is crucial for region-based object localization. Currently, there is still room for improvement, as one can see by the fact that the true maximum of the quality function, as found by ESS, often does not coincide with the expected ground truth. This aspect has recently been adressed in [31].

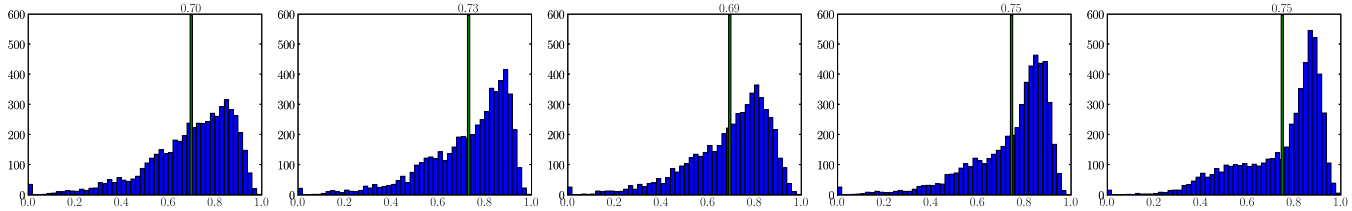
2) ESS vs. other Localization Systems: To compare ESS with other localization methods from the literature, we evalu-



(a) Histogram of relative number of evaluations $\frac{n^{ESS}}{n^{SW_i}}$ (log scale). In the blue region, sliding window required more evaluations than ESS. In the red region, ESS required more evaluations. The green bar indicates the mean ratio.



(b) Histogram of relative scores $\frac{f(y^{SW_i})}{f(y^{ESS})}$, where $y^{ESS} = \text{argmax}_{y \in \mathcal{Y}} f(y)$. The green bar indicates the mean ratio.



(c) Histogram of box overlap between regions y^{ESS} maximizing the quality function and the regions y^{SW_i} found by sliding window search: $\frac{\text{Area}(y^{ESS} \cap y^{SW_i})}{\text{Area}(y^{ESS} \cup y^{SW_i})}$. The green bar indicates the mean ratio.

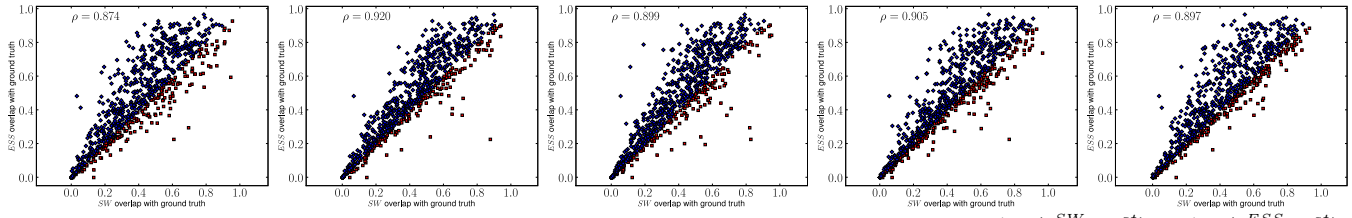
Fig. 4. Comparison of *ESS* against *sliding window* search, detecting classes `cat` and `dog` in all test images of PASCAL VOC 2006 (5372 images). From left to right, sliding window with five different parameter sets ($SW_1, \ldots, SW_5$, see Table I) are shown.

	maximal/minimum window size	**size-ratio**	**aspect ratios (AR)**	**stepsize x/y**
SW_1	full image to $32 \cdot (\sqrt{AR} \times \frac{1}{\sqrt{AR}})$	$\sqrt{2}$	2^l for $l \in \{-2, -1.5, \ldots, 2\}$	$1/16$ of window width/height
SW_2	full image to $32 \cdot (\sqrt{AR} \times \frac{1}{\sqrt{AR}})$	1.10	2^l for $l \in \{-2, -1.5, \ldots, 2\}$	$1/4$ of window width/height
SW_3	full image to $16 \cdot (\sqrt{AR} \times \frac{1}{\sqrt{AR}})$	1.05	2^l for $l \in \{-2, -1.5, \ldots, 2\}$	$1/2$ of window width/height
SW_4	full image to $20 \cdot (\sqrt{AR} \times \frac{1}{\sqrt{AR}})$	$\sqrt{\sqrt{2}}$	2^l for $l \in \{-2, -1.75, \ldots, 2\}$	$1/8$ of window width/height
SW_5	full image to $24 \cdot (\sqrt{AR} \times \frac{1}{\sqrt{AR}})$	1.10	2^l for $l \in \{-3, -2.75, \ldots, 3\}$	$1/8$ of window width/height

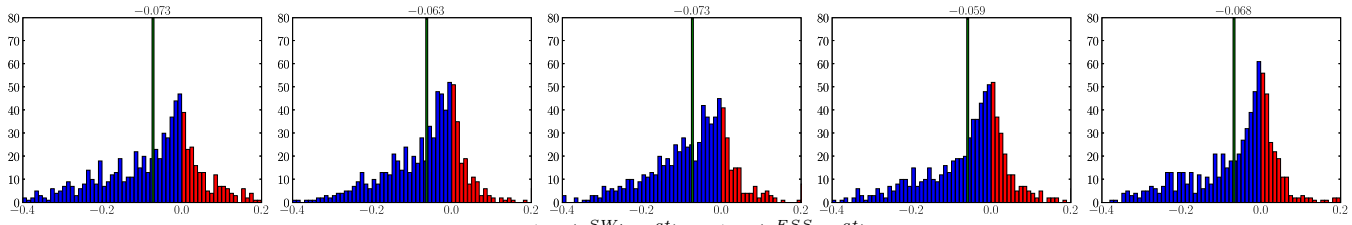
TABLE I
PARAMETERS OF SLIDING WINDOW SEARCHES FOR FIGURES 4 AND 5. **SIZE-RATIO** IS THE FACTOR USED FOR MULTI-SCALE DETECTIONS. THE PARAMETERS ARE CHOSEN SIMILAR TO TYPICAL METHODS FROM THE LITERATURE [10], [15], [30] AND ADAPTED TO ACHIEVE RUNTIMES COMPARABLE WITH ESS.

tate its absolute localization performance in the standardized setup of the PASCAL VOC dataset. First, we measure the system's performance in a pure localization task by applying ESS to only the test images that actually contain objects to be localized (*i.e.* cats or dogs). For each image we evaluate the best object location by the usual VOC method of scoring [32]: a detected bounding box is counted as a correct match if the area of overlap with the corresponding ground truth box is at least 50% of the area of their union. To each detection a confidence score is assigned that we set to the value of the quality function on the whole image. Figure 6 contains *precision–recall* plots of the results. The curves' rightmost points correspond to returning exactly one object per image. At this point of operation, approximately 55% of all cat bounding boxes returned are correct and 47% of all dog boxes. At the same time, we correctly localize 50% of all cats in the dataset and 42% of all dogs. Note that precision and recall differ, because images can contain more than one object instance. Moving along the curve to the left, only objects are included into the evaluation which have higher confidence scores assigned to them. This generally improves the localization precision.

As no other results on pure localization on the PASCAL VOC datasets have been published so far, we also performed the more common evaluation scenario of combined classification and localization. For this, the method is applied to all images of the test set, no matter if they contain the object to be searched for or not. It is the task of the algorithm to avoid false positives *e.g.* by assigning them a low confidence score. The performance is measured using the evaluation



(a) Scatter plot of overlaps between detected boxes for ESS y^{ESS} and sliding window y^{SW_i} with ground truth y^{gt}: $\frac{\text{Area}(y^{SW_i} \cap y^{gt})}{\text{Area}(y^{SW_i} \cup y^{gt})}$ vs. $\frac{\text{Area}(y^{ESS} \cap y^{gt})}{\text{Area}(y^{ESS} \cup y^{gt})}$. Boxes that ESS estimates better than SW_i are drawn in blue, others in red. ρ is the resulting correlation coefficient.



(b) Histogram of differences in overlap with ground truth: $\frac{\text{Area}(y^{SW_i} \cap y^{gt})}{\text{Area}(y^{SW_i} \cup y^{gt})} - \frac{\text{Area}(y^{ESS} \cap y^{gt})}{\text{Area}(y^{ESS} \cup y^{gt})}$. The bins for which ESS provides a better estimate than SW_i are drawn in blue, the others in red. The green bar indicates the mean difference.

Fig. 5. Comparison of *ESS* and *sliding window* search to *ground truth*, combined for `cat` and `dog` test images of PASCAL VOC 2006 (758 detections). From left to right, sliding window with five different parameter sets ($SW_1, \ldots, SW_5$, see Table I) are shown. ESS overall achieves higher overlap with ground truth than any of the sliding window methods.

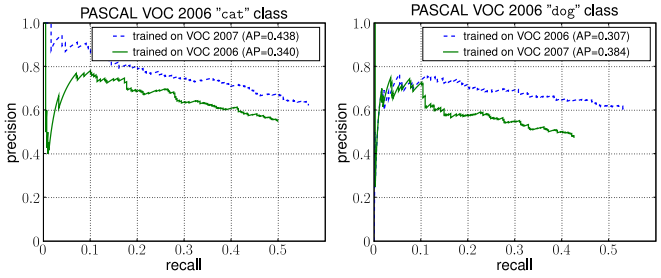


Fig. 6. Recall–Precision curves of ESS *bovw* localization for classes *cat* (left) and *dog* (right) of the VOC 2006 dataset. Training was performed either on VOC 2006 (solid line) or VOC 2007 (dashed).

method \ data set	cat	dog
ESS w/ bag-of-visual-words kernel	0.223	0.148
Viitaniemi/Laaksonen [33]	0.179	0.131
Shotton/Winn [32]	0.151	0.118

TABLE II
AVERAGE PRECISION (AP) SCORES ON THE PASCAL VOC 2006 DATASET. ESS OUTPERFORMS THE BEST PREVIOUSLY PUBLISHED RESULTS.

software provided in the PASCAL VOC challenges: from the *precision–recall* curves, the *average precision (AP)* measure is calculated, which is the average of the maximal precision within different intervals of recall, see [32] for details. Table II contains the results, showing that ESS improves over the best results that have been achieved in the VOC 2006 competition or in later publications. Note that the *AP* values in Table II are not comparable to the ones in Figure 6, since the experiments use different test sets.

B. PASCAL VOC 2007 challenge

An even larger and more challenging dataset than PASCAL VOC 2006 is the recent VOC 2007[4] dataset. It consists of 9,963 images with 24,640 object instances. We trained a system analogous to the one described above, now using the 2007 training and validation set, and let the system participate in the PASCAL VOC challenge 2007 on multi-view object localization. In this challenge, the participants did not have access to the ground truth of the test data, but had to submit their localization results, which were then evaluated by the organizers. This form of evaluation allows the comparison of different methods on a fair basis, making it less likely that the algorithms are tuned to the specific dataset.

With AP scores of 0.240 for cats and 0.162 for dogs, ESS clearly outperformed the other participants on these classes, with the runner-up scores being 0.132 for cats and 0.126 for dogs. By adopting a better image-based ranking algorithm, we were able improve the results to 0.331 and 0.177 respectively.

As an additional experiment, we took the system that had been trained on the 2007 *training* and *validation* data, and evaluated its performance on the 2006 *test* set. The results are included in Figure 6. The combination achieves higher recall and precision than the one trained on the 2006 data, showing that ESS with a bag-of-visual-words kernel generalizes well even across datasets and is able to make positive use of the larger number of training images available in the 2007 dataset.

VI. APPLICATION II: LOCALIZATION OF RIGID OBJECTS USING A SPATIAL PYRAMID KERNEL

For rigid and typically man-made object classes like cars or buildings, more informative representations have been developed than the *bag-of-visual-words* used in the previous section.

[4]http://www.pascal-network.org/challenges/VOC/voc2007/

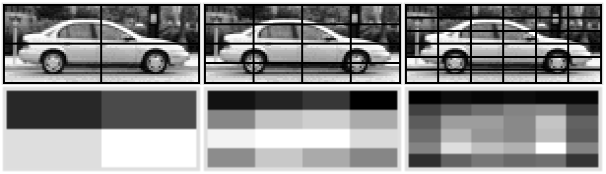


Fig. 7. Spatial Pyramid Weights. Top row: Example of a training image with its pyramid sectors for levels 2, 4 and 6. Bottom row: the energy of corresponding pyramid sector weights as learned by the SVM (normalized per level). Feature points in brighter regions in general have higher discriminative power.

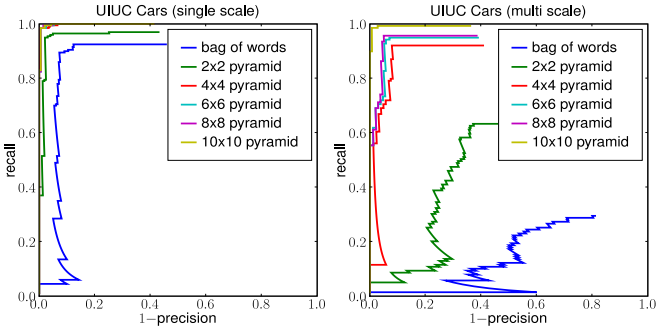


Fig. 8. Results on UIUC Cars Dataset (best viewed in color): $1-precision$ vs $recall$ curves for bag-of-features and different size spatial pyramids. The curves for single-scale detection (left) become nearly identical when the number of levels increases to 4×4 or higher. For the multi scale detection the curves do not saturate even up to a 10×10 grid.

In particular *hierarchical spatial pyramids* of features have recently proven very successful, *e.g.* in [16]. However, these previous approaches were usually limitated to few pyramid levels (typically 2 or 3) and required heuristic pruning. In this section, we will show that ESS allows efficient localization with pyramids as fine-grained as 10×10 grid cells without the risk of missing promising object locations.

A. UIUC Car dataset

As dataset we use the UIUC Car database[5], which is an example of a dataset with rigid object images (cars) from a single viewpoint. In total there are 1050 training images of fixed size 100×40 pixels. 550 of these show a car in side-view, the rest shows other scenes or parts of objects. There are two test sets of images with varying resolution. The first consists of 170 images containing 200 cars from a side view of size 100×40. The other test set consists of 107 images containing 139 cars in sizes between 89×36 and 212×85. We use the dataset in its original setup [34] where the task is pure localization. Ground truth annotation and evaluation software is provided as part of the dataset.

B. Experiments

From the UIUC Car training images, we extract SURF descriptors at different scales on a dense pixel grid and quantize them using a 1000 entry codebook that was generated from 50,000 randomly sampled descriptors. Since the training images already either exactly show a car or not at all, we do not require additional bounding box information and train the SVM with a hierarchical spatial pyramid kernel on the full training images. We vary the number of pyramid levels between $L = 1$ (*i.e.* a *bovw* without pyramid structure) and $L = 10$. The most fine-grain pyramid therefore uses all grids from 1×1 to 10×10, resulting in a total of 385 local histograms. Figure 7 shows an example image from the training set and the learned classifier weights from different pyramid levels, visualized by their total energy over the histogram bins. On the coarser levels, more weight is assigned to the lower half of the car region than to the upper half. On the finer pyramid levels, informative spatial regions are emphasized, *e.g.* the wheels become very discriminative whereas the top row and the bottom corners are almost ignored.

[5]http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/

method \data set	single scale	multi scale
ESS w/ 10×10 pyramid	1.5 %	1.4 %
ESS w/ 4×4 pyramid	1.5 %	7.9 %
ESS w/ bag-of-visual-words	10.0 %	71.2 %
Agarwal et al. [34]	23.5 %	60.4 %
Fergus et al. [35]	11.5 %	—
Leibe et al. [36]	2.5 %	5.0%
Fritz et al. [37]	11.4 %	12.2%
Mutch/Lowe [38]	0.04 %	9.4%

TABLE III
ERROR RATES ON UIUC CARS DATASET AT THE POINT OF EQUAL PRECISION AND RECALL.

At test time, we search for the best three car subimages in every test image as described in Section IV, and for each detection we use its quality score as confidence value. As it is common for the UIUC Car dataset, we evaluate the system's performance by a $1 - precision$ vs. $recall$ curve. Figure 8 shows the curves for several different pyramid levels. Table III contains error rates at the point where precision equals recall, comparing the results of ESS with the currently best published results. Note that the same dataset has also been used in many other setups, *e.g.* using different training sets or evaluation methods. Since the results of these are not comparable, we do not include them.

The table shows that localization with a flat *bovw*-kernel works acceptably for the single scale test set but poorly for multi scale. Using ESS with a finer spatial grid improves the error rates strongly, up to the level where the method clearly outperforms all previously published approaches on the multi scale dataset and all but one on the single scale dataset.

Note that for the single scale test set, a direct sliding window approach with fixed window size 100×40, would be computationally feasible as well. However, there is no advantage of this over ESS, as the latter requires even fewer classifier evaluations on average, and at the same time allows the application of the same learned model to the multi-scale situation without retraining.

VII. Application III: Image Part Retrieval using a χ^2-Distance Measure

ESS is applicable to more areas than only object localization. In the following, we apply ESS to the problem of *image part retrieval i.e.* to find images in a database based on queries that only have to match a part of the target image. This kind of search allows one not only to search for objects or persons, but also *e.g.* to find trademarked symbols on Internet image collections or in video archives.

A. χ^2-distance for content based image retrieval

We adopt a *query-by-example* framework similar to [39], [40], where the query is a region in an image, and we are interested in all frames or scenes in a video containing similar regions. For this, we use ESS to do a complete nearest-neighbor comparison between the query and all boxes in all database images. In contrast to previously proposed approaches, ESS allows the system to rely on arbitrary similarity measures between regions, not just on the number of co-occurring features. In our example, we choose the χ^2-distance that has shown good performance for histogram-based retrieval and classification tasks [41]. Specifically, we use the unnormalized variant χ^2_u, as this takes into account the total number of features and thereby the region size, which is desirable for the task at hand.

At first, we formulate the retrieval problem in an optimization framework, by defining the localized similarity between a query region q with *bovw*-histogram h^q and an image x as

$$\mathit{locsim}(x,q) = \max_{y \in \mathcal{Y}(x)} -\chi^2_u(h^q, h^y) \tag{19}$$

where h^y is the histogram for the subimage y of x and $\chi^2_u(h^q, h^y)$ is calculated as

$$\chi^2_u(h^q, h^y) = \sum_{k=1}^{K} \frac{(h^q_k - h^y_k)^2}{h^q_k + h^y_k}. \tag{20}$$

The retrieval task is now to identify the N images with highest localized similarity to q as well as the region within each of them that best matches the query.

Since *locsim* consists of a maximization over all subregions in an image, we can use ESS to calculate it. To construct the required bound, we use the construction for the χ^2-distance in Section III-C2, except that we do not have to normalize the histograms. In analogy to Equation (16), each summand in (20) is bounded from below by

$$\frac{(h^q_k - h^y_k)^2}{h^q_k + h^y_k} \geq \begin{cases} (h^q_k - \underline{h}^y_k)^2 / (h^q_k + \underline{h}^y_k) & \text{for } h^q_k < \underline{h}^y_k, \\ 0 & \text{for } \underline{h}^y_k \leq h^q_k \leq \overline{h}^y_k, \\ (h^q_k - \overline{h}^y_k)^2 / (h^q_k + \overline{h}^y_k) & \text{for } h^q_k > \overline{h}^y_k, \end{cases}$$

and their negative sum bounds $-\chi^2_u(h^q, h^y)$ from above.

B. Experiments

We show the performance of ESS in localized retrieval by applying it to 10242 keyframes of the full-feature movie *"Ferris Bueller's Day Off"*. Each frame is 880×416 pixels large. We extract SURF descriptors from keypoint locations, from a regular grid and from random localization and quantize them using a 1000 entry codebook. Each keyframe is therefore represented by 40,000–50,000 codebook entries.

For a given query region, multi-image ESS is used to return the 100 images containing the most similar regions. Figure 12 shows a query region and the search results. Since keyframes within the same scene or for repeated shots tend to look very similar, we show only one image per unique scene. ESS reliably identifies the *Red Wings* logo in different scenes regardless of strong background variations. Within the top 100 retrieval results there are no false positives.

In total, the search required $1.7 \cdot 10^8$ evaluations of the quality bound, that is approximately 170,000 per detection and 16,521 per image in the database. On images that were selected amongst the top 100, on average 57,000 evaluations were performed whereas on images that were not selected, only 16,100 evaluations were necessary. This shows that ESS sucessfully concentrated its effort on the promising images. In contrast, when running ESS on every keyframe separately, a total $1.04 \cdot 10^{11}$ evaluations were required. While for the top 100 images, the number of evaluations is identical to those for ESS, the images that were not selected on average required $1.03 \cdot 10^6$ per image, that is more than 600 times as many as in the case of joint ESS search. In fact, this number would even be higher, had we not restricted the maximal number of evaluations to 2 million per image.

Figure 9 shows the number of evaluations required to find the global maximum in each individual frame against the maximal score of the quality function (*locsim*). The 100 images with largest scores are marked in red, others in blue. As one can see, images with high similarity to the query require much fewer evaluations of the quality bound. This is because the quality function has a clear maximum in this case. The branch-and-bound search quickly identifies a general region of interest and then concentrates its computation on this region to find the exact maximum. For images that do not fill the query well, the quality function is typically rather flat, and many regions have quality scores similar to the optimal one. Consequently, many regions have to be checked before the algorithm can be sure that the global maximum has been identified.

The shape of the point cloud in Figure 9 allows some further reasoning. All images with a quality greater than -1750 required few calculations, indicating that the quality function found a clear maximum. We can therefore assume that all of these will contain the logo that is used as query.

In fact, checking the detection results of the per-frame search, the first false positive detection occurs at position 177 with *locsim* score of -1680, and in the range between -1680 and -1750, 8 more of 23 detections are false positives. In the range of -1750 and -1800, 37 out of 47 detections are from images not showing the query logo. With scores below -1800, the logo occurs only sporadically, and is often strongly distorted or truncated.

The same effect that good matches are easier to find than bad ones also has a strong effect on the total runtime when varying the number of images to return or images in the

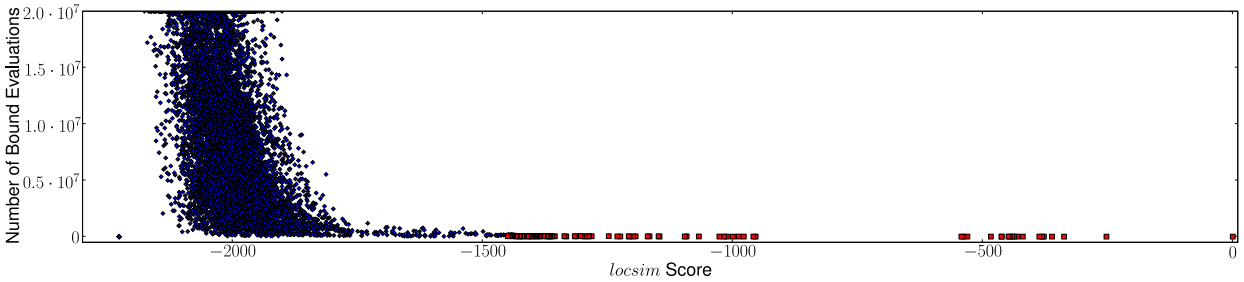


Fig. 9. Number of evaluations of the quality bound against the *locsim* value (Eq. (19)) for each image. Red squares indicate the top 100 images qualifying, blue diamonds the other images. Images with high *locsim* score require much fewer evaluations of the quality bound.

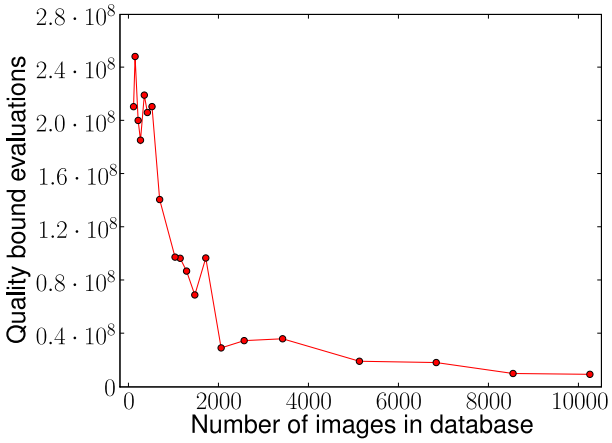


Fig. 10. Performance of multi-image ESS search for varying database sizes. With a larger database, the number of evaluations required to identify the 20 best matching images *decreases*.

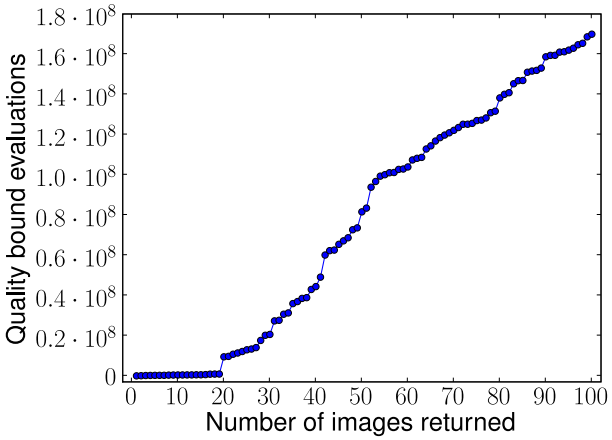


Fig. 11. Performance of multi-image ESS search for varying number of images to return. After an initial region of "easy" hits, the runtime is approximately linear in the number of output images.

database.

In Figure 10, we plot the total number of evaluations required to find the top 20 image regions for varying database sizes. For this, we reduce the database by subsampling it in regular intervals. Counterintuitively, the runtime *decreases* with more images in the database[6]. The reason for this is that a larger dataset is more likely to contain more clear matches to which ESS quickly converges.

Figure 11 shows the number of evaluations required by ESS with a joint priority queue to return different numbers of images from the dataset of 10242 keyframes. One can see that the method scales approximately linearly in the number of output images. For the first 19 hits, the slope is much smaller than on average, indicating that the search was especially easy. A check of the results shows that these detections are in fact near duplicates of the query region.

VIII. Conclusion

We have demonstrated how to perform fast object localization and localized retrieval with results equivalent to an exhaustive evaluation of a quality function over all rectangular regions in an image down to single pixel resolution. Sliding window approaches have the same goal, but in practice they have to resort to subsampling techniques and approximations to achieve a reasonable speed. In contrast to this, our method retains global optimality in its search, which guarantees that no maxima of the quality function are missed or misplaced.

The gain in speed and robustness allows the use of better local classifiers (*e.g.* SVM with spatial pyramid kernel, nearest neighbor with χ^2-distance), for which we demonstrated excellent results on the UIUC Cars, the PASCAL VOC 2006 dataset and in the VOC 2007 challenge. We also showed how to integrate additional properties, *e.g.* shape penalties, and how to search over large image collections in sublinear time.

In future work, we plan to study the applicability of ESS to further kernel-based classifiers. We are also working on extensions to other parametric shapes, like groups of boxes, circles and ellipses. These are often more desirable in applications of biological, medical or industrial machine vision, where high speed and performance guarantees are important quality factors as well.

Acknowledgments

This work was funded in part by the EU projects CLASS, IST 027978, and PerAct, EST 504321. We would like to thank Marcin Marszałek for sharing the confidence scores of his PASCAL VOC 2007 classifier with us.

[6]Since every image has to be inserted into the search queue, the method cannot be sublinear in the sense of computation complexity. However, the observed growth of runtimes is decreasing: the more images the database contains, the fewer operations are necessary in total to find the top N.

(a) Red Wings logo used as query

(b) Results of local search with χ^2-distance

Fig. 12. Image retrieval using a local χ^2 distance: the Red Wings logo (left) is used as a query region. b) shows the top results (one image per scene). The logo is detected in 9 different scenes. There are no false positives amongst the top 100 detected regions within 10242 keyframes.

REFERENCES

[1] T. M. Breuel, "Fast recognition using adaptive subdivisions of transformation space," in *CVPR*, 1992, pp. 445–451.

[2] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, 1993.

[3] M. Hagedoorn and R. C. Veltkamp, "Reliable and efficient pattern matching using an affine invariant metric," *IJCV*, vol. 31, no. 2-3, pp. 203–225, 1999.

[4] D. M. Mount, N. S. Netanyahu, and J. L. Moigne, "Efficient algorithms for robust feature matching," *Pattern Recognition*, vol. 32, no. 1, pp. 17–38, 1999.

[5] F. Jurie, "Solution of the simultaneous pose and correspondence problem using gaussian error model," *CVIU*, vol. 73, no. 3, pp. 357–373, 1999.

[6] C. F. Olson, "Locating geometric primitives by pruning the parameter space," *Pattern Recognition*, vol. 34, no. 6, pp. 1247–1256, 2001.

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

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *CVPR*, vol. 1, pp. 511–518, 2001.

[9] H. A. Rowley, S. Baluja, and T. Kanade, "Human face detection in visual scenes," in *NIPS*, 1996, pp. 875–881.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, pp. 886–893.

[11] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *PAMI*, vol. 30, pp. 36–51, 2008.

[12] O. Chum and A. Zisserman, "An exemplar model for learning object classes," in *CVPR*, 2007.

[13] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, no. 4, pp. 699–719, 1966.

[14] B. L. Fox, J. K. Lenstra, A. H. G. R. Kan, and L. E. Schrage, "Branching from the largest upper bound: Folklore and facts," *European Journal of Operational Research*, vol. 2, pp. 191–194, 1978.

[15] P. A. Viola and M. J. Jones, "Robust real-time face detection," *IJCV*, vol. 57, no. 2, pp. 137–154, 2004.

[16] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006, pp. 2169–2178.

[17] K. Grauman and T. Darrell, "The pyramid match kernel: Efficient learning with sets of features," *JMLR*, vol. 8, pp. 725–760, Apr. 2007.

[18] F. Schaffalitzky and A. Zisserman, "Viewpoint invariant texture matching and wide baseline stereo," in *ICCV*, 2001, pp. 636–643.

[19] S. Boughorbel, J.-P. Tarel, and N. Boujemaa, "Generalized histogram intersection kernel for image recognition," in *ICIP*, 2005, pp. 161–164.

[20] Swain and Ballard, "Color indexing," *IJCV*, vol. 7, 1991.

[21] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," in *ICIP*, 2003, pp. 513–516.

[22] F. M. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," in *CVPR*, 2005, pp. 829–836.

[23] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *CVPR*, 2008.

[24] R. E. Moore, *Interval Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1966.

[25] Hickey, Ju, and V. Emden, "Interval arithmetic: From principles to implementation," *JACM: Journal of the ACM*, vol. 48, 2001.

[26] T. M. Breuel, "On the use of interval arithmetic in geometric branch and bound algorithms," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1375–1384, Jun. 2003.

[27] T. Yeh and T. Darrell, "Fast concurrent object localization and recognition," in *CVPR*, 2009.

[28] B. Gendron and T. G. Crainic, "Parallel branch-and-bound algorithms: survey and synthesis," *Operations Research*, vol. 42, pp. 1042–1066, 1994.

[29] H. Bay, T. Tuytelaars, and L. J. Van Gool, "SURF: speeded up robust features," in *ECCV*, 2006, pp. 404–417.

[30] I. Laptev, "Improving object detection with boosted histograms," *Image and Vision Computing*, vol. 27, no. 5, pp. 535–544, 2009.

[31] M. B. Blaschko and C. H. Lampert, "Learning to localize objects by structured output regression," in *ECCV*, 2008.

[32] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool, "The PASCAL visual object classes challenge 2006 (VOC2006) results," http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf, 2006.

[33] V. Viitaniemi and J. Laaksonen, "Techniques for still image scene classification and object detection," in *ICANN (2)*, vol. 4132, 2006, pp. 35–44.

[34] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *PAMI*, vol. 26, no. 11, pp. 1475–1490, 2004.

[35] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *CVPR*, 2003, pp. 264–271.

[36] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *IJCV*, vol. 77, no. 1-3, pp. 259–289, May 2008.

[37] M. Fritz, B. Leibe, B. Caputo, and B. Schiele, "Integrating representative and discriminative models for object category detection," in *ICCV*, 2005, pp. 1363–1370.

[38] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," in *CVPR*, 2006, pp. 11–18.

[39] N.-S. Chang and K.-S. Fu, "Query-by-pictorial-example," *IEEE Transactions on Software Engineering*, vol. 6, no. 6, pp. 519–524, 1980.

[40] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *ICCV*, 2003, pp. 1470–1477.

[41] B. Schiele and J. L. Crowley, "Object recognition using multidimensional receptive field histograms," in *ECCV*, 1996, pp. I:610–619.