Detecting Objects in Large Image Collections and Videos by *Efficient Subimage Retrieval*

Christoph H. Lampert
Max Planck Institute for Biological Cybernetics, Tübingen, Germany
christoph.lampert@tuebingen.mpg.de

Abstract

We study the task of detecting the occurrence of objects in large image collections or in videos, a problem that combines aspects of content based image retrieval *and* object localization*. While most previous approaches are either limited to special kinds of queries, or do not scale to large image sets, we propose a new method,* efficient subimage retrieval (ESR)*, which is at the same time very flexible and very efficient. Relying on a two-layered branch-and-bound setup, ESR performs object-based image retrieval in sets of 100,000 or more images within seconds. An extensive evaluation on several datasets shows that ESR is not only very fast, but it also achieves detection accuracies that are on par with or superior to previously published methods for object-based image retrieval.*

1. Introduction

In this work, we study the problem of *object-based image retrieval*, that is the detection of objects in very large image collections or in videos. In particular, we are interested in methods that allow queries to be defined as image regions themselves, making the problem a combination of *content based image retrieval* and *object localization*.

From an image retrieval user's point of view, the possibility of object-based instead of image based queries is a clear benefit, because the relevance of images will not be effected by changes in image viewpoint or background clutter anymore, see Figure 1 for an illustration. However, most image retrieval systems internally rely on purely global image representations, and they are not able to handle queries that match only small regions within the images. Object localization methods, on the other hand, are in principle capable of answering the question of whether and where an object occurs in an image, but they are typically overburdened when having to deal with very many candidate images, because they do not scale well in terms of runtime and memory usage. Consequently, most existing systems for object-based image retrieval either achieve high detection accuracy, but work only for small image collections, or they can handle large set of candidate images, but are limited in the types of local queries that they can answer.

Figure 1. *Object-based image retrieval.* Left: The users marks an object in an image by a bounding region. Right: The system returns images from a database that show the query object.

In this work, we introduce a new method for object-based image retrieval, *efficient subimage retrieval (ESR)*, that is proficient in both aspects. It finds the images and local regions best matching a query, irrespective of background clutter and how small the query region is, and it does so efficiently, requiring only seconds or less to answer queries about data sets with 100,000 images or more.

The rest of the paper is organized as follows: Section 2 introduces ESR by deriving it from the efficient subwindow search (ESS) procedure for object localization. Based on an analysis of ESS' shortcomings, we show how to modify and improve the underlying branch-and-bound optimization to allow large-scale retrieval tasks. In Section 3, we recall object-based image retrieval methods from the literature and discuss their relation to ESR. Section 4 shows experimental results and analyzes ESR's performance in terms of detection quality and speed. Finally, Section 5 summarizes the paper and hints on directions for future work.

2. Efficient Subimage Retrieval

Efficient subimage retrieval (ESR) is a two-stage, globally optimal branch-and-bound procedure that can be seen as an extension of the single-stage *efficient subwindow search (ESS)* procedure [8]. We therefore start by summarizing the main idea and notation of ESS.

2.1. Branch-and-Bound Object Localization

Let I be an image, for which we assume that a visual word representation has been precomputed, such that we can represent arbitrary image regions R by their bag of vi-

EFFICIENT SUBWINDOW SEARCH$(I, f, \hat{f})$
1 $P_{\text{loc}} \leftarrow$ empty *sets-of-regions* priority queue
2 $\mathcal{R} \leftarrow \mathcal{R}_{all}(I)$
3 **repeat**
4 split $\mathcal{R} \rightarrow \mathcal{R}_1 \dot{\cup} \mathcal{R}_2$
5 push $\mathcal{R}_1$ into P_{loc} with score $\hat{f}(\mathcal{R}_1)$
6 push $\mathcal{R}_2$ into P_{loc} with score $\hat{f}(\mathcal{R}_2)$
7 pop $\mathcal{R} \leftarrow top(P_{\text{loc}})$
8 **until** $\mathcal{R} = \{R\}$ for a single image region $R \subset I$
9 **return** R

Figure 2. *Efficient Subwindow Search* [8] in pseudocode notation. The input consists of an image I and a quality function f with bounding function $\hat{f}$. $\mathcal{R}_{all}(I)$ denotes the set of all candidate subimages of I. The algorithm outputs the region $R \subset I$ maximizing the quality function f over all elements of $\mathcal{R}_{all}(I)$.

sual word histograms h^R. By $\mathcal{R}_{all}(I)$ we denote the set of all rectangular subregions in I. Then ESS can be described as a geometric branch-and-bound optimization that finds the region $R \in \mathcal{R}_{all}(I)$ maximizing a given quality function $f : \mathcal{R}_{all}(I) \rightarrow \mathbb{R}$. Table 2 gives pseudocode for ESS.

ESS uses an internal representation of *sets of boxes* $\mathcal{R}$ by uncertainty intervals for their left, top, right and bottom coordinates. This induces a canonical *branching* step (line 4) by splitting the interval of largest uncertainty into halves. For the *bounding* step, one constructs a *quality bounding function* $\hat{f} : \mathbb{P}(\mathcal{R}_{all}(I)) \rightarrow \mathbb{R}$, where $\mathbb{P}$ denotes the power set operation, that fulfills the conditions

$$\hat{f}(\mathcal{R}) \geq \max_{R \in \mathcal{R}} f(R), \quad \text{and} \quad \hat{f}(\{R\}) = f(R), \quad (1)$$

for any $R \in \mathcal{R}_{all}$ and any $\mathcal{R} \subset \mathcal{R}_{all}(I)$ that can occur during the search. Quality bounding functions for linear SVMs and for nearest neighbor classifiers based on the χ^2-distance were derived in [8]. For these, ESS is able to find the global maximum of f with a worst-case computational complexity of $\mathcal{O}(n^4)$ for $n \times n$ images. In typical situations, ESS is much faster than the worst-case analysis indicates, showing rather $\mathcal{O}(n^2)$ runtime. Because of the data dependency of the branch-and-bound search, there are, however, also situations where ESS requires a lot of iterations to converge, especially if f does not have a distinct global maximum.

A naïve approach to object-based retrieval would apply ESS to every candidate image, sorting the results by their quality and returning the best matches. However, this setup is computationally infeasible for large setups: even if processing a single image takes only milliseconds, it would be several minutes before a dataset with 100,000 images would be fully processed. Clearly, this is an unacceptable delay for the typically interactive application scenarios of object-based image retrieval.

In order to overcome the linear dependency on the number of images, a modification to ESS is proposed in [8] that inserts the start states $\mathcal{R}_{all}(I_i)$ for all image $I_1, \ldots, I_n$ into the priority queue before starting the branch-and-bound search. This results in $70\times$ speedup compared to the linear setup, with an average retrieval time of 2 seconds per detection when searching through 10,243 images. However, for truly large-scale object-based retrieval tasks, ESS even in this variant is not well suited, because it still has several limitations caused by the fact that it was originally derived to work on single images at a time. In the following, we analyze these limitations and propose a new algorithm *efficient subimage retrieval (ESR)* that overcomes them. Specifically, ESR improves over ESS in the following aspects:

Computation complexity: for ESS to initialize P_{loc} with $\mathcal{R}_{all}(I_1), \ldots, \mathcal{R}_{all}(I_n)$, it has to process every image at least once. We propose a *two layered* branch-and-bound search that can achieve sublinear runtime with respect to the number of images.

Memory usage: ESS needs to keep all image representation in main memory simultaneously, thereby limiting the number of images that it can search over. We show how to overcome this by a *load on demand* strategy.

Growth of P_{loc}: for difficult queries the priority queue in ESS can grow uncontrollably into tens of millions of entries, requiring hundreds of MB of RAM. We introduce a *new representation for box sets* that allows constraints on the size of the search box. We show that by searching with boxes of fixed size, the problem of the growing P_{loc} becomes irrelevant.

Lack of sparsity: the quality bound for the χ^2-distance proposed in [8] does not respect sparsity. Even for query histograms with few non-zero entries, it has to store and process all bins. We derive bounds for *five classical quality function* from image retrieval that respect the sparsity of the query and at the same time achieve good detection accuracy.

2.2. Branch-and-Bound Subimage Retrieval

In this section, we explain how to "fix" ESS to make it applicable to realistically sized image retrieval problems. Figure 3 shows pseudocode for the resulting ESR algorithm.

2.2.1 Quality Functions for Image Retrieval

The most easily solvable problem of ESS is its choice of quality function. In [8], the χ^2 distance $\chi^2(h^Q, h^R) = \sum_{k=1}^{K} \frac{(h_k^Q - h_k^R)^2}{h_k^Q + h_k^R}$ is used because of the known good performance of the χ^2-kernel for object classification tasks. However, the bound derived for it is suboptimal for image retrieval tasks, because in such a time-critical application, one would prefer a quality function that does not require many floating point divisions and that can benefit from the expected *sparsity* of the query histogram h^Q. Several classical similarity measures from the image retrieval literature comply with these conditions while also achieving good detection accuracy. In this work, we propose using the *histogram intersection (HI)*, the *normalized histogram inter-*

REGION-B&B($P_{\text{loc}}, \hat{f}$)
1 pop $[I, \mathcal{R}] \leftarrow top(P_{\text{loc}})$
2 **if** $\mathcal{R} = \{R\}$ for a single region $R \subset I$
3 **then** remove all states $[I, *]$ from P_{loc}
4 **return** (I, R)
5 **else** split $\mathcal{R} \rightarrow \mathcal{R}_1 \dot{\cup} \mathcal{R}_2$
6 push $\mathcal{R}_1$ into P_{loc} with score $\hat{f}(\mathcal{R}_1)$
7 push $\mathcal{R}_2$ into P_{loc} with score $\hat{f}(\mathcal{R}_2)$
8 **return** $\emptyset$

IMAGE-B&B($P_{\text{img}}, P_{\text{loc}}, \tilde{f}$)
9 pop $\mathcal{I} \leftarrow top(P_{\text{img}})$
10 **if** $\mathcal{I} = \{I\}$ for a single image $I \in \mathcal{I}_{all}$
11 **then** push $[I, \mathcal{R}_{all}(I)]$ into P_{loc} with score $\tilde{f}(\{I\})$
12 **else** split $\mathcal{I} \rightarrow \mathcal{I}_1 \dot{\cup} \mathcal{I}_2$
13 push $\mathcal{I}_1$ into P_{img} with score $\tilde{f}(\mathcal{I}_1)$
14 push $\mathcal{I}_2$ into P_{img} with score $\tilde{f}(\mathcal{I}_2)$
15 **return** $\emptyset$

EFFICIENT SUBIMAGE RETRIEVAL($\mathcal{I}_{all}, f, \hat{f}, \tilde{f}, m$)
16 $P_{\text{img}} \leftarrow$ empty *sets-of-images* priority queue
17 push $\mathcal{I}_{all}$ into P_{img} with score $\tilde{f}(\mathcal{I}_{all})$
18 $P_{\text{loc}} \leftarrow$ empty *sets-of-regions* priority queue
19 **for** $j \leftarrow 1$ **to** m
20 **do repeat**
21 **if** $top(P_{\text{img}}).score > top(P_{\text{loc}}).score$
22 **then** $res =$ IMAGE-B&B($P_{\text{img}}, P_{\text{loc}}, \tilde{f}$)
23 **else** $res =$ REGION-B&B($P_{\text{loc}}, \hat{f}$)
24 **until** $res \neq \emptyset$
25 $J_j, R_j \leftarrow res$
26 **return** $(J_1, R_1), \ldots, (J_m, R_m)$

Figure 3. The *Efficient Subimage Retrieval* algorithm in pseudocode notation. The input consists of a collection of images $\mathcal{I}_{all} = \{I_1, \ldots, I_n\}$, a quality function f with bounding functions $\hat{f}$ and $\tilde{f}$ (see text), and a target number of output regions. The algorithm outputs a sorted list of m images $J_1, \ldots, J_m$ and regions $R_j \subset J_j$ that maximize the quality function f over all regions within all images of $\mathcal{I}_{all}$.

section (NHI), the Euclidean *dot product (dot)* or the *cosine measure (cos)*, see Table 1 for their definitions. For completeness, we have also included the *Bhattacharya coefficient*, which does respect sparsity, but is much slower than the other measures, because it requires the calculation of many square roots. Note that *HI* and *dot* are only useful for situations where we have constraints on the region size, because they are not normalized, see Section 2.2.2.

To use the proposed quality functions as part of a branch-and-bound search, we require *quality bounds* in the sense of Equation (1). Because all quality functions are histogram-based, these bounds are rather straight-forward to compute: to bound f over a set of regions $\mathcal{R}$, we define $\overline{h}^{\mathcal{R}}$ as the histogram of the union of all regions in $\mathcal{R}$, and $\underline{h}^{\mathcal{R}}$ be the histogram of their intersection, where the latter can also be empty. As all quality functions in Table 1 decompose into sums of elementary function of the individual histogram bins, their bounds can be found by bounding each summand separately in terms of the bins of $\overline{h}^{\mathcal{R}}$ and $\underline{h}^{\mathcal{R}}$. For quality functions based on L^p normalized histograms, we make use of the fact that $\|h^R\|_p \geq \|\underline{h}^{\mathcal{R}}\|_p$ for all $R \in \mathcal{R}$. The resulting expression are given in the third column of Table 1.

2.2.2 Constraints on the Search Region

A fundamental problem of ESS is the rapid growth of the priority queue P_{loc}. Even when the number of images is not infeasibly high, the search can easily run out of memory, especially when searching for a rare object. The reason for this strong growth of P_{loc} is an unfortunate combination of the relatively loose quality bounds with the extremely large search space consisting of billions of candidate regions for each image in the dataset.

While one could think of applying pruning techniques, this would introduce the risk of missing detections. In ESR, we therefore follow a different route: instead of optimizing f approximately over the space of all regions, we keep the guarantee of global optimality but restrict the space of candidate regions. In particular, we drop the property of searching for rectangular regions of arbitrary size and aspect ratio, because in a query-by-example image retrieval system, the query region provides us with a strong enough prior of the expected shape and size of the object. Incorporating this prior information not only strongly reduces the search space that the branch-and-bound search has to cover, but it also implicitly regularizes the quality function, because only "reasonably shaped" box regions are returned.

The ESS representation of box sets by their left, right, top and bottom coordinate does not allow for an easy integration of size constraints. Therefore, we introduce a new parametrization of box set $\mathcal{R} = [X, Y, W, H]$, where $X = [\overline{x}, \underline{x}]$ and $Y = [\overline{y}, \underline{y}]$ are interval coordinates for the box center and $W = [\overline{w}, \underline{w}]$ and $H = [\overline{h}, \underline{h}]$ are intervals limiting the width and height respectively. Note that we still have closed form expressions for calculating the union and intersection over the elements of $\mathcal{R}$:

$$\cup\mathcal{R} = [\ \tfrac{1}{2}(\overline{x} + \underline{x}),\ \tfrac{1}{2}(\overline{y} + \underline{y}),\ \overline{w}+\overline{x}-\underline{x},\ \overline{h}+\overline{y}-\underline{y}], \quad (2)$$

$$\cap\mathcal{R} = [\ \tfrac{1}{2}(\overline{x} + \underline{x}),\ \tfrac{1}{2}(\overline{y} + \underline{y}),\ \underline{w}-\overline{x}+\underline{x},\ \underline{h}-\overline{y}+\underline{y}]. \quad (3)$$

A particularly useful property of this representation is that we can search over all windows of a fixed size by setting $\underline{w} = \overline{w}$ and $\underline{h} = \overline{h}$, thus turning the four-dimensional branch-and-bound search effectively into a two-dimensional search. This results in a vast reduction in search iterations and at the same time renders the memory requirements of P_{loc} negligible. The experiments also show that due to the use of scale invariant feature descriptors we do not completely lose the ability to detect objects of other scales. Finally, fixed size search windows allow the use of the simpler *dot* and *HI* quality measure that have degenerate behavior if regions can be of arbitrary size.

symbol	quality function f	region-level quality bound $\hat{f}$	image level quality bound $\tilde{f}$
$dot(h^Q, h^R)$	$\sum_k h_k^Q h_k^R$	$\sum_k h_k^Q \overline{h}_k^{\mathcal{R}}$	$\sum_k h_k^Q \overline{h}_k^{\mathcal{I}}$
$HI(h^Q, h^R)$	$\sum_k \min(h_k^Q, h_k^R)$	$\sum_k \min(h_k^Q, \overline{h}_k^{\mathcal{R}})$	$\sum_k \min(h_k^Q, \overline{h}_k^{\mathcal{I}})$
$cos(h^Q, h^R)$	$\frac{1}{\|h^Q\|_2 \|h^R\|_2} \sum_k h_k^Q h_k^R$	$\frac{1}{\|h^Q\|_2 \|\underline{h}^{\mathcal{R}}\|_2^*} \sum_k h_k^Q \overline{h}_k^{\mathcal{R}}$	$\frac{1}{\|h^Q\|_2} \sum_{\{k\,:\,\overline{h}_k^{\mathcal{I}}>0\}} h_k^Q$
$NHI(h^Q, h^R)$	$\sum_k \min\left(\frac{h_k^Q}{\|h^Q\|_1}, \frac{h_k^R}{\|h^R\|_1}\right)$	$\sum_k \min\left(\frac{h_k^Q}{\|h^Q\|_1}, \frac{\overline{h}_k^{\mathcal{R}}}{\|\underline{h}^{\mathcal{R}}\|_1^*}\right)$	$\frac{1}{\|h^Q\|_1} \sum_{\{k\,:\,\overline{h}_k^{\mathcal{I}}>0\}} h_k^Q$
$\rho_{bhatt}(h^Q, h^R)$	$\frac{1}{\sqrt{\|h^Q\|_1 \|h^R\|_1}} \sum_k \sqrt{h_k^Q h_k^R}$	$\frac{1}{\sqrt{\|h^Q\|_1 \|\underline{h}^{\mathcal{R}}\|_1^*}} \sum_k \sqrt{h_k^Q \overline{h}_k^{\mathcal{R}}}$	$\frac{1}{\sqrt{\|h^Q\|_1}} \sum_{\{k\,:\,\overline{h}_k^{\mathcal{I}}>0\}} \sqrt{h_k^Q}$

Table 1. Examples of quality functions suitable for *Efficient Subimage Retrieval* with their region- and image-based bounding functions: *histogram intersection (HI)*, *normalized histogram intersection (NHI)*, *dot-product (dot)*, *cosine measure (cos)* and *Bhattacharya coefficient (ρ_{bhatt})*. $\|.\|_p$ denotes the L^p-norm for $p = 1, 2$, and $\|.\|_p^* := \max(1, \|.\|_p)$. See Sections 2.2.1 and 2.2.3 for the remaining notation.

2.2.3 Branch-and-Bound over Image Sets

An image retrieval method will not be truly scalable if it has to perform a linear scan through all images for each query. In ESS, this is particularly apparent when searching only for very few detections, because then it is the step of inserting all images into P_{loc} that dominates the overall search time. ESR gets around this linear scan by introducing an image level branch-and-bound process, based on a second priority queue P_{img}. In each iteration, ESR checks whether the upper bounds of the image queue of P_{loc} or P_{img} are higher (Table 3, line 21), and executes one step in the more promising queue (line 22, 23).

The localization step (line 1–8) works as it does for ESS: it extracts the most promising set of regions (line 1), splits it (line 5) and reinserts the resulting states into P_{loc} (line 6,7). Because we require only one detection per image, whenever the algorithm has identified a detection R in an image I, it removes all states for candidate regions within I from the priority queue (line 3) before returning R. The image level branch-and-bound procedure (line 9–15) has the analog steps of extracting the top state (line 9), splitting it (line 12) and reinserting the parts (line 13,14), except that if an optimum I is identified, it is transferred to the localization queue for further processing (line 11).

The branch-and-bound search in the image domain requires an *image level quality bound*, *i.e.* a function $\tilde{f} : \mathbb{P}(\mathcal{I}_{all}) \to \mathbb{R}$ that fulfills

$$\tilde{f}(\mathcal{I}) \geq \max_{I \in \mathcal{I}} \hat{f}(I), \tag{4}$$

for any $\mathcal{I} \subset \mathcal{I}_{all}$, where $\hat{f}(I)$ is a short-hand notation for $\hat{f}(\mathcal{R}_{all}(I))$. In contrast to Equation (1), no condition of equality between $\hat{f}$ and $\tilde{f}$ for single element sets $\mathcal{I}$ is necessary, but ESR will correctly identify the object detections even for a trivial choice like $\tilde{f}(\mathcal{I}) := \infty$. However, a too loose bound $\tilde{f}$ renders the image-based priority queue superfluous, because very large values of $\tilde{f}$ will cause all images to be moved from P_{img} to P_{loc} during the first iterations of the algorithm. One should therefore try to have at least $\tilde{f}(\{I\})$ close to $\hat{f}(I)$, as this will ensure that only images with promising $\hat{f}$ values will show up in P_{loc}. By this construction one also gets away without a pruning step in P_{img}, which would require *a priori* knowledge of the minimal quality score.

For histogram based quality functions suitable bounds $\tilde{f}$ can be constructed in similar way to $\hat{f}$: for any set of images $\mathcal{I}$, we set, under a slight abuse of notation,

$$\tilde{f}(\mathcal{I}) := \hat{f}(h^Q, \overline{h}^{\mathcal{I}}) \tag{5}$$

for the histogram $\overline{h}^{\mathcal{I}}$ defined by $\overline{h}_k^{\mathcal{I}} = \max_{I \in \mathcal{I}} h_k^I$ and $\underline{h}_k^{\mathcal{I}} = 0$, and writing h^I for $h^{\mathcal{R}_{all}(I)}$. Table 1 lists the resulting expressions in column 4.

The time evaluate Equation (5) is independent of the size of $\mathcal{I}$ if we precompute the query independent histograms $\overline{h}^{\mathcal{I}}$. In contrast, the naïve bound $\tilde{f}(\mathcal{I}) = \max_{I \in \mathcal{I}} f(h^Q, h^I)$ is potentially tighter, but already calculating the initial $\tilde{f}(\mathcal{I}_{all})$ spoils the chances of sublinear runtime complexity.

How good the bounds $\tilde{f}$ as given by Equation (5) actually are depends on the elements of $\mathcal{I}$. The more homogeneous the histograms h^I are for $I \in \mathcal{I}$, the tighter $\tilde{f}$ is. Consequently, if the optimal region is actually contained in $\mathcal{I}$, fewer evaluation within irrelevant images will be required. In the following, we introduce two possible splitting methods, one for *videos* and one for arbitrary *image sets*.

Scene-Based Splits of Videos. In order to split the set of frames of a video sequence, we can make use of the prior knowledge that the frames have a temporal order, and that neighboring frames typically have similar bag of visual words histograms. Consequently, we represent sets of frames by intervals over the time axis and split them in every iteration at their center point. Better bounds, however, are obtained by taking into account that a typical video contains hundreds or thousands of scene changes. Because the bag of word histograms can change significantly at these points, using scene changes as natural splitting points leads

to more homogeneous histogram sets. We implement this, by first identifying all positions in the video at which a scene change occurs. During the search, we split a set of frames $\mathcal{I} = \{I_{low}, \ldots, I_{high}\}$ always at the scene change closest to the center of the interval, or, if $\mathcal{I}$ does not contain a scene change, we split at the center frame $I_{\lfloor \frac{1}{2}(low+high) \rfloor}$ itself.

Because the resulting hierarchical partitioning of $\mathcal{I}_{all}$ is independent of the query, we can precompute the resulting histograms $h^{\mathcal{I}}$ and consider them parts of the dataset.

Kernel Vector Quantization Clusters. For arbitrary image collections, the concept of a *scene* does not exist, and splitting along the image index will likely not be better than splitting randomly. Instead, we construct the intended splits by *hierarchical clustering* using *Kernel Vector Quantization (KVQ)* [20]. In contrast to clustering methods used previously to speed up image retrieval [16, 18], in particular to k*-means* and *agglomerative clustering*, the KVQ algorithms find clusters such that the maximal distance between any two elements within a cluster is guaranteed not to exceed a user-definable threshold. Because KVQ can work with any distance metric, it provides a powerful tool for finding clusters that are homogeneous with respect to the quality measure f. For this, we first observe that all quality functions f in Table 1 are positive definite Mercer kernels [13], and their induced distance functions

$$d_f(h, h') = \sqrt{f(h,h) + f(h',h') - 2f(h,h')}. \quad (6)$$

are therefore proper metrics. To construct a hierarchical partitioning of $\mathcal{I}_{all}$, we start with the full dataset. In each following step, we calculate the largest distance between any two elements in the current image set. Running KVQ with the square root of this distance as threshold, we are ensured to obtain clusters with significantly smaller radius than the current one. Note that, because we fix the maximal distortion instead of the number of prototypes, the number of clusters found can vary. This is no problem, however, as the IMAGE-B&B routine of ESR is easily adapted to handle splits into more than two parts. As in the situation of videos, the clustering obtained is independent of the query. It can therefore be precomputed and the resulting aggregated histograms can be stored as part of the dataset.

2.2.4 Reducing Memory Usage by *Lazy Loads*

ESR as described so far has already the capability of performing object-based image retrieval in sublinear runtime. However, its memory footprint would still prevent it from working with truly large data sets, as can be seen from the following rough calculation. A typical visual word representation of an image consists of 1000–5000 local descriptors. For each one, we have to store its (x, y) location and its codebook id. Even in a packed representation, *e.g.* using *short integers* values, each image requires 6–30 KB of random access memory. For a video or dataset with 100,000 images, already up to 3 GB of RAM are required, which is near or beyond the limit of what typical workstation offer.

In ESR, we mitigate this problem by a *lazy load* strategy, making use of the best first property of the branch-and-bound search. We first observe that it is only the region-based quality function $\hat{f}$ that requires information on where in an image the feature points are located. The image based $\tilde{f}$ can be calculated based only on the accumulated histograms $h^{\mathcal{I}}$. Consequently, we can delay the loading of the feature point representation for an image until it is moved from P_{img} to P_{loc} (line 11), which might well be never, namely if $\tilde{f}$ indicates that it is not relevant for the query. While in principle it is also possible that P_{img} runs empty, *i.e.* all images lie simultaneously in P_{loc}, we have never observed such behavior during our experiments.

2.3. Further Improvements and Extensions

One can imagine several further extensions and improvements that do not affect the search step of ESR itself and can therefore be included optionally. Note that some of them are approximations, and by adopting them one loses the guarantee that ESR detects only the globally best regions.

Stopword removal: We can prefilter the set of visual words, removing the most and/or least frequent ones [17]. Though in our own experience, this step did not improve the detection accuracy, it also did not reduce it, thus making stopword removal at least benefit in the sense that it reduces in the number of feature points we have to store and process.

Inverted indices: To benefit from the sparsity of the query histogram, we can build *inverted local maps*, similarly to *inverted indices* [17]. For every image and visual word, we store all locations at which corresponding feature points occur. By pre-sorting the location along one of their coordinates, we can calculate h_k^R faster than by a linear scan through all feature points.

Image filtering: We can speed up the search and reduce the maximal memory requirements by limiting the number of images that are allowed to be moved from P_{pic} to P_{loc}.

Output reranking: Post-filtering operations, *e.g.* based on context or geometry, are easily added, because ESR returns not only the relevant images but also the region coordinates.

Search time limits: ESR is an *anytime* algorithm. If we interrupt the search at any time, we still have the guarantee that all detections up to this point were the optimal ones.

Result diversity: We can enforce *diversity* in the result by modifying the REGION-B&B procedure. For every detection $[I, R]$, we remove not only all search states containing I from the priority queues, but also all states corresponding to all frames within the same scene or cluster.

Feature combination: As a histogram based technique, ESR can naturally combine different feature types, *e.g.* in form of additional histogram dimensions.

Feature weighting: All quality functions of Table 1 can be

made into weighted, *e.g*. *tf-idf*, variants. All quality functions are additive, so the weights propagate into the bounds.
Multiple queries: To search for multiple queries simultaneously, *e.g*. multiple regions or different region sizes, it suffices to insert all start states into the initial priority queue.
User feedback: Because ESR returns the "best" boxes first, users can immediately see how promising their current search is. User feedback can be integrated on-the-fly.

3. Related Work

Research in the area of content based image retrieval goes back as far as the late 1970s, see *e.g*. [9, 15] for overviews. Only a small part of the method developed, however, can perform object-based local queries. We divided those into three categories: methods that decompose the images into collections of smaller units, methods that augments global representations, and methods that rely on object localization techniques.

Decomposition method, *e.g*. [1, 4, 12] partition the image into discriminative image segments, sometimes called *blobs*, and allow queries to the individual part or combinations instead of just to the image as a whole. However, because only elements of the preconstructed image decomposition are accessible for comparison to the query and retrieval, these methods are typically limited in what kind of queries they can answer successfully. *Augmented representation* rely on a preprocessing of all images to mark all object occurrences that might later by queried for. Despite their limitations, augmentation is a powerful technique when the set of possible queries is limited, *e.g*. only faces [2, 10] or human poses [3]. *Object localization* based techniques are more flexible that the previous ones, because they can in principle identify any part of the image as relevant for a query. Early methods used color histogram [19, 22] and searched the entire image for the best matching region, typically applying heuristics to speed up the search. Similar setups were also applied to deformable object templates [6] or SIFT descriptors [23]. The drawback of localization based systems are usually their lack of scalability, because each object localization step is computationally costly and the images are typically processed sequentially.

Surprisingly, the currently most successful systems for object-based image retrieval in large image collections do not really perform local searches at all, but rather only combine flexible local image representations with image level retrieval techniques. In [5] every image is decomposed into a set of local descriptors. To search for a query region, the images are ranked based on nearest neighbors lookups between all descriptors of the query and the images. A more scalable system is proposed in [7] using *locally sensitive hashing* to match descriptors. The *Video Google* system [14] also represents images by local descriptors, but it accumulates them into bag-of-visual-words feature. Retrieval is done by calculating the *cosine* measure between the local query histograms and the histograms of every image in the dataset. Such local-to-global comparison are efficient to computable, but they do not reflect the local nature of the query. Therefore, post-processing operations are applied that re-ranked the highest scoring images according to a second quality measure, taking into account e.g. the local geometry. An improved system that can also handle much larger dataset is proposed in [11]. The authors choose a very large codebook with over one million elements, thereby strongly reducing the number of false positive matches on the global level. The highest scoring frames are reranked based on a RANSAC procedure.

ESR can be seen as a fusion of the localization based methods with the global-to-local methods. Crucial differences, however, are that ESR does not search linearly over all images, and it also does not use the comparison between local query and full images as hard cut-off to discard image. Instead, the quality bound $\tilde{f}$ guides a search for promising images to which the localization procedure is applied.

4. Experimental Evaluation

In the following, we show the performance of ESR for several task of object-based image retrieval from videos and image collections. For this, we have implemented ESR in C++ with a small Python GUI for query selection[1]. Except where indicated otherwise, all experiments were performed on a Dell workstation with 3.4 GHz Intel CPU and 3 GB of RAM. All reported runtimes are measured *user time*, in order to exclude the overhead of disk access at startup.

In order to put the method into perspective with previous work, we first analyze the quality of the proposed quality functions by performing local-to-global queries on the *OxfordBuildings dataset*[2], thereby copying the procedure described in [11]. Table 2 shows the results where retrieval quality is defined as the average precision (AP) measure over 55 predefined local queries. One can see that all quality functions of Table 1 have comparable and good performance, clearly better than the χ^2-distances. Using the same features representation, the authors of [11] report results slightly better than ours, which we contribute to their use of a stopword table and *tf-idf* weighting. RANSAC based post-processing causes a further marginal improvement.

To show ESR's performance on the retrieval of local regions, we applied it to several videos and the Caltech256 image dataset[3]. In each case we extracted up to 1000 rgSIFT [21] descriptors per image and quantized them using codebooks of 1000 (for (a),(b),(d)) or 512 (for (c),(e)) visual words. Table 4 shows the results in graphical and nu-

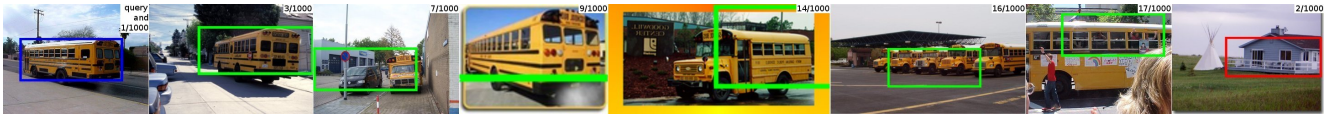
[1] Source code is available at `http://www.christoph-lampert.de`.
[2] `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`
[3] `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`

(a) *Ferris* (10,243 keyframes from [8]) T: $(<0.01/0.1/0.9/35.2)$, E: $(0.021/3.9/26/282)$, Q: $(1.0/0.66/0.41/0.19)$, S: $(99.98/78.0/47.5/18.7)$

(b) *Scrubs* (25,134 frames). T: $(2.2/2.8/10.1/95.8)$, E: $(118/135/204/561)$, Q: $(0.40/0.39/0.37/0.28)$, S: $(35.4/33.9/29.5/16.1)$

(c) *Simpsons* (124,820 frames). T: $(2.0/18.9/103/2281)$, E: $(88/706/1330/4214)$, Q: $(0.70/0.49/0.43/0.34)$, S: $(50.7/24.7/20.1/15.0)$

(d) *Soccer* (191,568 frames). T: $(<0.01/2.5/16/237)$, E: $(0.095/105/215/520)$, Q: $(1.00/0.74/0.70/0.65)$, S: $(99.96/70.8/65.2/59.1)$



(e) *Caltech256* (30,607 images) T: $(<0.01/2.3/11/242)$, E: $(0.018/101/264/5958)$, Q: $(1.00/0.43/0.37/0.30)$, S: $(99.99/64.2/54.9/43.5)$

Figure 4. Graphical and numerical results of object-based image retrieval with ESR. Each row shows from left to right: query image, example detections (suppressing near-duplicates), highest scoring false detection. Numerically, we report the following quantities for 1, 10, 100 and 1000 detections: (T) total runtime [in seconds], (E) total number of evaluations of quality function $\hat{f}$ [in 1000], (Q) value of quality function f, (S) image level sparsity (*i.e.* images not moved to P_{loc}) [in percent] .

χ^2	χ^2_{norm}	*dot*	*HI*	*cos*	*NHI*	ρ_{bhatt}	*tf-idf cos* [11]
0.03	0.34	0.56	0.54	0.61	0.57	0.60	0.62 *(0.65)*

Table 2. Average precision of different quality function on the *OxfordBuildings* dataset. χ^2 fails, and χ^2_{norm} performs weakly, because they are ill-suited to the very sparse representation. All quality functions from Table 1 perform comparably with the best result in [11], where a *tf-idf* weighted cosine measure after stopword removal is used. The italicized score is achieved by RANSAC post-processing.

merical form. Because of space limitation we only report the results of the NHI measure. In summary, the other quality functions perform as one would expect them to: *HI* and *dot* are faster than *NHI* but achieve slightly lower accuracy. *cos* is comparable in quality and speed with *NHI*. ρ_{bhatt} is of comparable quality as well, but much slower.

Experiment (a) is set up as a copy of the retrieval experiment in [8], using only the keyframes of the full-length movie *Ferris Bueller's Day Off*. To achieve comparable timing results, we also use a PC with 2.4 GHz CPU as described there. Our results show that ESR achieves superior retrieval accuracy (no false detections within the first 100 detections compared to 4) at over 200× the detection speed (0.9s for 100 detections instead of 200s). The other experiments show further aspects of interest: (b) uses the same query as (a), but the search is performed over all frames of an episode of the TV series *Scrubs* containing a variant of the same *Red Wings* logo. This is a more difficult setup than the previous, because the image material of query and database come from different sources and therefore have different characteristics, e.g. in terms of image noise. Nevertheless, ESR sucessfully identified the logo and also copes well with object occurrences of different size and occlusions, despite the use of a fixed size query window. In (c), we applied ESR to *The Simpsons Movie*. Because cartoon drawings contain few textured regions Retrieval in this setup is difficult task for SIFT-based methods, This results in low values of the quality function for all matches and relatively high retrieval times, but the visual quality of the detections is still good. This example also illustrates that query regions for ESR do not have to be rectangular, because only the resulting bag of visual words histogram is required to perform a search. Experiment (d) illustrates an application of potential commercial interest: we detect, how often and when a certain *FUJIFILM* advertisement was visible during a soccer match. Note that this would be a very difficult problem for methods that work with global representations, because most frames look very similar, and the 102×16 pixel query region corresponds

to only 0.3% of the 1000×568 pixel images. Retrieval on Caltech256 dataset (e) shows the limitation of the query-by-example setup chosen. While ESR does detect images with similar regions, it is not designed to generalize between object instances and it cannot perform reliable object category classification based on its single object example. Overall, the experiments show that ESR is capable of doing high quality, highly efficient, large-scale object-based image retrieval. Particularly impressive results are achieved when the dataset contained a perfect subimage match for the query (experiments (a),(d),(e)). These are identified within milliseconds, requiring less than a hundred evaluation of $\hat{f}$.

From our experience, the main two factor for ESR's high performance are the reduction of the search space by relying on fixed size queries, and the replacement of the linear search over all images by the prioritized image queue P_{img}. The image-level branch-and-bound step is beneficial only for very large image collections and if a good pre-clustering of the images is available. Otherwise, inserting all frames directly into P_{img} is an easier and equally efficient setup.

5. Conclusions

In this work we have introduced *efficient subimage retrieval*, a generalization of efficient subwindow search that allows globally optimal object-based image retrieval for large image datasets. Our main contributions were the introduction of a new box set parametrization that is suitable for retrieval tasks with search windows of fixed or constrained size, a two layer branch-and-bound setup that keeps non-promising images in a separate queue, thereby saving time and memory, and the derivation of quality bounds for several classical quality functions for image retrieval. In an experimental evaluation we were able to show excellent detection performance in terms of speed and accuracy: typically, it takes ESR only seconds to find the regions best matching a query within a sets of over 100,000 images. Because ESR is constructed in a modular way, it opens the doors for many different directions of future work. We have hinted on some of them in Section 2.3.

Acknowledgments

We thank the EC project CLASS, IST 027978, for funding.

References

[1] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *PAMI*, 24(8), 2002.

[2] T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval: An experimental comparison. *Information Retrieval*, 11(2), 2008.

[3] V. Ferrari, M. Marín, and A. Zisserman. Pose Search: retrieving people using their pose. In *CVPR*, 2009.

[4] D. A. Forsyth, J. Malik, M. M. Fleck, H. Greenspan, T. K. Leung, S. Belongie, C. Carson, and C. Bregler. Finding pictures of objects in large collections of images. In *Object Representation in Computer Vision II*. Springer, 1996.

[5] V. Gouet and N. Boujemaa. Object-based queries using color points of interest. In *IEEE CBAIVL*, 2001.

[6] A. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *PAMI*, 18(3), 1996.

[7] Y. Ke, R. Sukthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, 2004.

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

[9] M. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM TOMCCAP*, 2(1):1–19, 2006.

[10] A. Martinez. Face image retrieval using HMMs. In *IEEE CBAIVL*, 1999.

[11] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[12] R. Rahmani, S. Goldman, H. Zhang, S. Cholleti, and J. Fritts. Localized content-based image retrieval. *PAMI*, 30(11), 2008.

[13] B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, 2002.

[14] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *PAMI*, 31(4), 2009.

[15] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *PAMI*, 22(12), 2000.

[16] B. Song, M. Kim, and J. Ra. A fast multiresolution feature matching algorithm for exhaustive search in large image databases. *IEEE TCSVT*, 11(5), 2001.

[17] D. Squire, W. Müller, H. Müller, and T. Pun. Content-based query of image databases: Inspirations from text retrieval. *Pattern Recognition Letters*, 21(13-14), 2000.

[18] D. Stan and I. Sethi. Image retrieval using a hierarchy of clusters. In *ICAPR*. Springer, 2001.

[19] M. Swain and D. Ballard. Color indexing. *IJCV*, 7(1):11–32, 1991.

[20] M. Tipping and B. Schölkopf. A kernel approach for vector quantization with guaranteed distortion bounds. In *AISTATS*, 2007.

[21] K. van de Sande, T. Gevers, and C. Snoek. Evaluation of color descriptors for object and scene recognition. In *CVPR*, 2008.

[22] V. Vinod and H. Murase. Focused color intersection with efficient searching for object extraction. *Pattern Recognition*, 30(10), 1997.

[23] W. Wu and J. Yang. Object fingerprints for content analysis with applications to street landmark localization. In *ACM Multimedia*, 2008.