Towards automatic bounding box annotations from weakly labeled images

Christian X. Ries · Fabian Richter · Rainer Lienhart

Abstract In this work we discuss the problem of automatically determining bounding box annotations for objects in images whereas we only assume weak labeling in the form of global image labels. We therefore are only given a set of positive images all containing at least one instance of a desired object and a negative set of images which represent background. Our goal is then to determine the locations of the object instances within the positive images by bounding boxes. We also describe and analyze a method for automatic bounding box annotation which consists of two major steps. First, we apply a statistical model for determining visual features which are likely to be indicative for the respective object class. Based on these feature models we infer preliminary estimations for bounding boxes. Second, we use a CCCP training algorithm for latent structured SVM in order to improve the initial estimations by using them as initializations for latent variables modeling the optimal bounding box positions. We evaluate our approach on three publicly available datasets.

Keywords Automatic annotation · Weakly labeled data · Statistical feature model · Visual features · Image analysis

1 Introduction

Successful approaches to object detection or image retrieval often require a set of training images which come with manual annotations indicating the locations of instances of a wanted object class within the training images. In this context, a common means of annotation is manually drawing tight rectangular regions of interest also called bounding boxes. Manually annotating bounding boxes, however, is a tedious task. Also, in some cases it is not desirable or even not possible. For example, consider the case of adult images where manual annotation is hardly acceptable.

C. X. Ries (✉) · F. Richter · R. Lienhart
Universität Augsburg, Universitätsstr. 6a, 86650 Augsburg, Germany
e-mail: chrisries13@web.de

In this work we discuss the problem of automatic object annotation. Automatic object annotation denotes the task of automatically finding locations of instances of a given object class within images without exploiting any knowledge beyond global binary image labels. Thus, the goal is reducing manually drawing bounding boxes to only providing sets of positive and negative images and obtaining bounding box estimations without further human supervision as illustrated in Fig. 1.

This work constitutes a detailed description of our previous work [16] and [17] on the topic of automatic object annotation. It also concisely summarizes [18]. We also introduce a number of enhancements, i.e. multi-object annotation and a latent structured training algorithm. Besides a thorough discussion of the problem, we provide an in-depth explanation of the methods used in [17] and explain implementation decisions in detail. We also describe further experiments.

This article is organized as follows. First, we present previous and related work in the next section. In Section 3, we give a problem definition for the task of automatic object annotation. Afterwards, in Section 4 we explain an approach to automatic object annotation based on a straight forward statistical feature model. The next section describes two implementations of the statistical model, namely a color model and a visual words model. In Section 6, we propose a CCCP-based structured latent learning algorithm for improving the bounding boxes obtained from the initial model. Evaluations and discussion of results are given in Sections 7 and 8. Finally, Section 9 concludes this work.

2 Previous and related work

In this section we briefly review our previous work on the topic of automatic object annotation and the basic methods it applies. Also, we name a few approaches on similar problems.

2.1 Previous work

This work is mainly based on previous work on the topic of automatic annotations, i.e. [16] and [17].

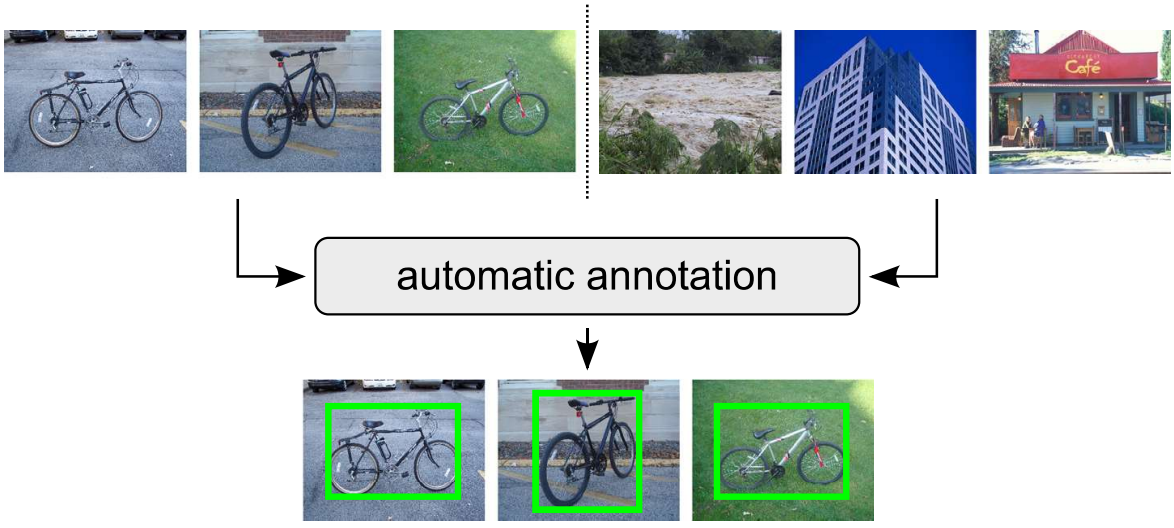

Fig. 1 Visualization of automatic object annotation problem: We are given a positive (*upper left*) and a negative (*upper right*) image set and find bounding boxes for the object instances within the positive images

In [16], the special problem of creating a color model for an unknown object class is discussed for the case when only global image labels are available and hence building a color model based on object pixels is not possible. The basic idea is to use statistics which can be derived from global labels in order to deduce a discriminative color model. The discriminative color model is then enhanced by a flood fill algorithm which results in image regions which are likely to consist of pixels with colors indicative for the wanted object. From these regions, a histogram-based color model is computed as suggested by Jones and Rehg [11]. As a result, pixels in test images can then be classified as either object pixels or background pixels.

The follow-up work [17] deals with the problem of automatic object annotation as defined in Section 3. Thus, for a given set of images which all contain at least one instance of an object class, the goal of [17] is to determine the locations of these object instances without further manual annotation. For this purpose, the same color model as in [16] is used for determining pixels which are likely to belong to an object instance. In addition to the color model, a visual words model is used which is implemented analogously to the color model. As local visual features, Histograms of Oriented Gradients (HOG) [5] cells are used. The combination of colors and visual features yield bounding boxes which are candidates for object locations. Based on these candidates, an SVM model is iteratively trained and evaluated on the positive images in order to improve the bounding box locations.

In this work summarizing [18], we present an approach which is basically an enhancement of [17]. We also begin with a statistical feature model. However, we suggest a method for unifying models for multiple features such that our basic statistical model becomes generic and independent from the actual feature used. Also, we introduce a heuristic for estimating multiple bounding boxes per image as opposed to [17] which only detects a single instance for each image. Furthermore, we propose using a latent structured learning algorithm for improving the bounding boxes obtained from the initial model, as opposed to the custom SVM algorithm used in [17]. Also note that in [17] additional knowledge about the aspect ratio of the wanted object is exploited in contrast to our approach where the aspect ratio is modeled by a latent variable. Our learning algorithm is an implementation of the Convex-Concave Procedure (CCCP) algorithm [27] and is inspired by the approaches of Yu and Joachims [26] and Zhu et al. [29]. The optimization component of the CCCP algorithm we use is based on Joachims et al. [10] and Tsochantaridis et al. [25].

2.2 Work on related problems

In general, the problem of automatic object annotation belongs in the category of learning from weakly labeled data, since we try to learn an object model from only global image labels. It is therefore similar to Multiple Instance Learning (MII) [6] as for example used in [13, 28]. Another interesting work in this context is provided by Siva et al. [22] who discuss the benefit of mining negative sets for learning from weakly labeled data.

Recently, learning from unlabeled or weakly labeled data has attracted much interest in the research community. In the following, we shortly present a selection of works on related problems.

A very similar problem to automatic annotation is determining generic objects within images, i.e. finding bounding boxes around image regions which are likely to show an object. An approach to this problem is proposed by Alexe et al. [1, 2] who measure the "Objectness" of bounding boxes. We also use their approach for comparison and thus discuss it in more detail in our evaluation chapter.

Another similar problem is determining discriminative segment annotations from weakly labeled video as suggested by Tang et al. [24]. Here, a weakly labeled video (i.e. labeled by a tag) is segmented into candidate regions. Then, regions with consistent visual features are identified across multiple video frames. They also use a large set of negative video frames for determining such consistent segments.

Chen et al. [4], search for synthetic training examples in weakly labeled videos based on a few manually labeled training examples. Despite a different scenario and different assumptions, their approach tackles a problem related to automatically annotating objects.

3 Problem definition

A main concern of this work is to define the problem of automatic object annotation in a general manner.

Let P be a set of images which share a common object. More precisely, each image in P shows at least one instance of a given object class. Thus P is a set of *positive* images. Also, we are given a negative set N which contains images which do not show the wanted object (except for neglible noise).

The task of automatic annotation then requires that we find the locations of said object instances in all positive images by devising a tight bounding box around them. Thus, a set of rectangles $\hat{R}_I$ is assigned to each image I in P whereas the rectangles must have a large overlap with bounding boxes one would obtain from manual annotation. Therefore, manual bounding boxes are required for evaluation but may not be exploited for automatic annotation.

As overlap criterion, we use the widely acknowledged intersection divided by union measure for rectangles which was for instance used in the VOC challenge [7]. For evaluation, we hence use manual ground truth bounding boxes R_I for each positive image I. The quality of the j-th bounding box estimation $\hat{r}_{j,I} \in \hat{R}_I$ in image I is then determined by its overlap with the one rectangle $r_{k,I} \in R_I$ which has the best overlap with $\hat{r}_{j,I}$ and is not already assigned to another estimation with an even higher overlap.

Since we aim at automatic annotation, we assume no further knowledge about the wanted object class besides the fact that each positive image has at least one object instance. Therefore, an approach to automatic object annotation must be weakly supervised, i.e. unsupervised except for providing a positive set of images. Also, we assume we are given a negative image set which does not contain any object instances.

Note that providing a positive set of images is nowadays a considerably easier task than drawing bounding box manually, since there are many ways for obtaining images with an assigned verbal tag from the Internet. Even though downloading images by tag usually yields many irrelevant (i.e. negative) images, browsing through a set of images and manually deleting irrelevant ones is still less work than annotating bounding boxes. Also, obtaining a negative set of background images is straightforward for many object classes, since often "random" scenes are acceptable. Therefore, we think that our problem definition describes a realistic scenario.

From our problem definition, two major assumptions arise about the wanted object class. First, it must obviously be possible to distinguish object instances from background with regards to the set of features used. In other words, the intra-class variance of the respective object class must be smaller than the variance in the backgrounds of the positive images. Second, the negative set must contain a fairly representative collection of background scenes. If the background set is heavily biased towards certain scenes, we cannot

deduce which visual features are likely to be present in positive images only due to their regular appearance in background.

4 Statistical feature model

Our first approach to automatic annotation is based on the intuition that visual features exist which are indicative for the wanted object class. The method is an enhanced version of [17]. The main difference is the dynamic threshold which replaces feature-specific thresholds of [17].

4.1 Confidence interval

An indicative visual feature is a feature f from a set F of discrete visual features which appears regularly on the wanted object without appearing as often in random background. We also refer to indicative features as positive features while the remaining features are negative features.

Note that without annotations, we do not know whether f appears on the wanted object or in the background of a positive image. In fact, the number of images which contain feature f at least once, however, is an upper bound to the number of images which contain f only *because* the wanted object is present (which characterizes an indicative feature). Let therefore $f(I) = 1$ denote the event that image I contains feature f at least once (on the object or in the background) and let $P_P(f(I) = 1)$ be the relative number of images in positive set P for which $f(I) = 1$. Analogously, let $P_N(f(I) = 1)$ be the relative number of negative images containing f. Now $P_N(f(I) = 1)$ serves as our expected value for observing feature f in a set of images if f is *not* an indicative feature. If feature f is observed in the positive images less often, i.e. if $P_N(f(I) = 1) > P_P(f(I) = 1)$ we can safely assume that f is a negative feature. Even if f was characteristic for the wanted object, it could not help distinguishing the object from background. In the opposite case, i.e. $P_N(f(I) = 1) \leq P_P(f(I) = 1)$ we cannot directly deduce that f is a positive feature since the fact that we observed f more frequently in P might be due to the fact that the background of the positive images is not completely random or set N is not quite representative. In fact, experiments show that there are many cases where negative features appear more frequently in P than in N.

We therefore analyze our confidence that the observation of relative frequency $P_P(f(I) = 1)$ implies that f is still a negative feature despite being more frequently present in the positive set. We borrow our confidence model from sample statistics: If we observe a property $f(I)$ among a relative number $P_N(f(I) = 1)$ of individuals of a set N, the probability of observing the same property $P_P(f(I) = 1)$ in another (sample) set is governed by a binomial distribution which we approximate by the following Gaussian:

$$n^{-1} N(x_f; \mu_f, \sigma_f^2) \tag{1}$$

where $n = |P|$, $x_f = P_P(f(I) = 1)$, and $\mu_f = P_N(f(I) = 1)$. The variance of this distribution is $\sigma_f^2 = p_f \cdot (1 - \mu_f) n^{-1}$. On this distribution we can define our confidence that a feature is a negative feature by means of a one-sided confidence interval and devise a binary decision function $c(f) : F \to \{0, 1\}$ as

$$c(f) = \delta(x_f \notin [0, p_f + \theta_f]). \tag{2}$$

The bound of a confidence interval is usually expressed in terms of the standard deviation in order to dynamically adjust to different sizes n and observations μ_f (which both determine the standard deviation σ_f):

$$\theta_f = z_F \cdot \sigma_f \tag{3}$$

This equation involves a constant z_F which depends on feature type F. However, since we do not assume any knowledge about how suitable feature F is for the object class at hand, we need to select z_F dynamically as explained in the following section.

4.2 Dynamic threshold selection

For bounding our confidence in feature f being not indicative for the wanted object class, we cannot simply introduce an empirical constant. The reason is that the degree to which a feature type F is suitable for a given object class is unknown and may strongly vary. For example consider a color feature applied on the object class of cars. Since cars come in a large number of different colors which are not unusual in background scenes, color is not indicative for cars. If the object class is for instance a certain brand logo, however, color may be very useful for locating object instances.

For features which are highly suitable for the given object class, we want a large confidence interval resulting in a small number of positive features, since usually a few very distinctive features are sufficient to locate an object. As explained below, we aim at combining multiple features by merging them using a logical AND operator. As a consequence, features which are not suitable should yield a relative small confidence interval and thus a large number of positive features.

In order to determine how suitable feature type F is for the wanted object class, we examine the "most indicative" feature which is the feature with the smallest value of the normal distribution of (1). To account for noise, we actually use the median of the 5 features with the 5 smallest values in practice. Let f^* denote this reference feature. Now we can determine a value z_F^* for z_F in (3) such that only f^* becomes a positive feature. Finally, we set our confidence bound relative to this value z_F^*:

$$\theta_f = \alpha z_F^* \cdot \sigma_f^2 \tag{4}$$

Note that constant $\alpha \in [0, 1]$ does not depend on the actual feature type but scales the feature-dependent z_F^* which we can compute without any previous knowledge about the feature type or the object class. Empirically, we determine that $\alpha = 0.75$ is a reasonable value.

Now feature types with a strong reference feature f^* will obtain a large confidence interval and a small set of positive features and vice versa, i.e. we obtain a large set of positive featues for feature types with weak reference feature. This is beneficial for our method which merges multiple features as explained in the following section.

4.3 Combining multiple features

Let F_{pos} be the set of positive features for feature type F, i.e. the set of features which lie outside our confidence interval defined in (2):

$$F_{pos} = \{f \in F | c(f) = 1\} \tag{5}$$

A visual feature can usually be linked to one or multiple pixel locations. Since the relations between features and pixels depends on the actual feature type, they are explained

Fig. 2 Examples for positive pixels based on two features and resulting bounding box estimations. *Left*: Positive pixels based on color $B_c(I)$ (*green*), on HOG features $B_{hog}(I)$ (*blue*), and their intersection $B(I)$ (*cyan*). *Center*: Gaussian Mixture models selected by the multi-box heuristic. *Right*: resulting bounding boxes. Best viewed in color

in the respective sections below. Now let $B_F(I)$ be the set of positive pixels obtained via positive feature set F_{pos}.

For combining multiple features in order to obtain a single set $B(I)$ of positive pixels, we simply intersect the sets of positive pixels over all feature types F:

$$B(I) = \bigcap_F B_F(I) \tag{6}$$

Note that the intersection is equivalent to a logical AND on every pixel over all feature types. As a consequence, false positive pixels of one feature do not harm the overall result if another feature is more accurate. Therefore we want large sets F_{pos} for features which are not suitable for the wanted object class which is the reason for our method of determining the confidence bound by dynamic threshold as described above.

The left column of Fig. 2 shows a few examples for two sets of positive pixels based on a color model (green pixels) and a HOG model (blue pixels), and their intersection (cyan pixels). In the second column the intersection is visualized by white pixels, i.e. the white pixels correspond to the cyan pixels of the left column. It can be seen that if one feature "over detects" the object, the respective other feature dominates the overall result. Also, Fig. 3 gives examples for positive features found by our feature model.

4.4 Bounding boxes

The positive set of pixels can be viewed as a binary map of positive pixels, i.e. a two-dimensional distribution of positive pixels. In order to determine bounding boxes we now attempt to fit Gaussian mixture models into this two-dimensional distribution using a standard EM algorithm. Since we do not know how many object instances image I contains, we apply an heuristic which fits Gaussian mixture models with $k \in \{1, ..., 5\}$ mixture components into $B(I)$. The fitting method (a standard EM algorithm) implicitly estimates the likelihood of the Gaussian model parameters which we then use to determine a value for k. For each k we compare this likelihood to the likelihood we obtained for the previous value $k-1$ which yields the increase (i.e. the difference) $i(k)$ of likelihood for k mixture components. We finally pick the value of k for which we observe the largest increase $i(k)$ of the likelihood of the model parameters of the mixture model (with an empirical lower bound of minimum increase). We also add a bias factor in order to slightly prefer larger k, i.e. it is only required that $i(k) \geq 0.8 \cdot i(k-1)$ for k to be selected.

After determining k Gaussian mixture components, each component yields one bounding box, i.e. one estimation for the location of an object instance. The bounding box is obtained as a box around the mixture component's mean. Width and height of the bounding box correspond to the component's standard deviation in the respective direction, multiplied with a empirically determined enlargement factor of 1.6.

Since this procedure may lead to unreasonable bounding boxes, we reject boxes with aspect ratios which contradict common sense thresholds. Also, pairs of bounding boxes which overlap by at least 0.1 are merged into a single bounding box surrounding both original boxes.

As a final post-processing step we apply Hough voting in order to merge pairwise bounding box configurations which occur in many positive images into single bounding boxes. Consider for example an object which consists of two parts which are detected by our method but disconnected due to non-indicative features in-between the two parts. We then obtain similar pairs of bounding boxes with respect to their relative locations and aspect ratios among a large number of positive images. By Hough voting such regularities can be identified and merged into single boxes with acceptable computational effort. We build a histogram over the aforementioned pair-wise features (relative locations and aspect ratios) and determine which configurations appear in a considerable number of positive images. Then, the respective pairs are merged into their convex hull.

In the center column of Fig. 2, some example results of the multi-bounding box heuristic are shown. The selected Gaussian mixture model is visualized by ellipses around 1.6 standard deviations in each direction. The right column shows the resulting bounding boxes along with white boxes which are manual annotations. The second example also shows a mixture component which is rejected during post-processing due to its extreme aspect ratio.

5 Implementations of statistical model

For our statistical model, we use two instantiations with two different visual features: Color and gradient based local features, i.e. HOG features [5]. In the following we give implementation details on both features.

5.1 Color

As our first feature, we use colors. Colors are arguably the most straightforward visual features. Still, they are discriminative for a relatively large number of object classes, since many objects occur in a limited number of different color combinations.

Our set of features F_c is hence a set of discretized colors. We conduct experiments analogous to [16] for determining the number of bins and color space for the color model. For our model, the HSV color space and 8 bins per color channel perform best (even though the difference to other color spaces and numbers of bins is relatively small). We determine a set $F'_{c,pos}$ of positive colors as explained in the previous section based on (5).

Following [16], we enhance the color model by using a flood-fill algorithm. In short, the positive colors from $F'_{c,pos}$ are used to determine seed pixels in the positive images. Starting at these seed pixels, similarly colored regions are determined which enables us to apply the region-based histogram approach explained in [11] which yields our final set $F_{c,pos}$ of positive colors.

The relation between positive features and positive pixels is trivial since each pixel has exactly one color.

5.2 Local visual features

As our second feature we use Histograms of Oriented Gradients (HOG) [5]. Since HOG features model gradient information, they are often complementary to color features to a certain degree. Recall that our feature set must be discrete. Therefore, we cluster a large number of HOG features into a set F_h of 10,000 discrete features represented by the respective cluster id. This procedure is commonly known as building a visual vocabulary [23].

As HOG cell size we use 8×8 pixels and we extract features over multiple scaled versions of the respective image (our scaling factor is $2^{-0.25}$ and we use 13 scales). Note that the color model already provides bounding boxes indicating (often too large) regions of interest which in almost all cases contain the wanted object. We therefore exploit this prior information for removing some background by creating a single bounding box per image based on the color model. We then extract the HOG features from the positive images only within these bounding boxes over all scales. The HOG implementation we use follows [8]. Also, we enrich our local feature representation by concatenating 2×2 HOG cells into a single feature vector before clustering.

Again, we apply the statistic model explained above in order to obtain a set of positive features $F_{h,pos}$. Positive pixels are then determined by considering all pixels positive which lay under a positive HOG cell which we re-scale into the original image.

5.3 Feature examples

In Fig. 2 we show a few examples for the positive pixel sets $B_c(I)$ and $B_h(I)$ obtained from positive colors ($F_{c,pos}$) and HOG features ($F_{h,pos}$), respectively. We also show the set of pixels $B(I)$ we obtain from intersecting $B_c(I)$ and $B_h(I)$.

The first example shows a situation where the HOG model works better than the color model. In the second example, the color model has less false detections and thus dominates the intersection. However, some false positive pixels of the color model are still removed by the HOG model. In the third example both features remove different background areas.

Fig. 3 Examples for positive features. *Left*: Four example training images (out of 70) for the class "DHL" from FlickrLogos-32 dataset [20]. *Center*: All positive colors determined by our statistical model with f_c^* on top (for visualization, we use RGB color space instead of HSV). *Right*: Three examples for positive visual words, each represented by four image patches with f_{hog}^* on top. Best viewed in color

Figure 3 shows examples for positive features. On the left, a few training images are shown. Next to the images, all positive colors are displayed beginning with the most indicative color f_c^* on top. On the right, three positive visual words are visualized, each by four example patches falling into the respective feature cluster. Again, the uppermost feature corresponds to the most indicative visual word f_{hog}^*. For the sake of visualization, here f_c^* and f_{hog}^* are the actual best features (with lowest values for the distribution of (1)) instead of the median of the top 5 features.

6 CCCP algorithm

In many cases, our initial model yields rectangle estimations which contain the wanted object. However, they often also contain relatively large background areas. In this section we explain how we improve our estimations by a latent structured learning algorithm.

Note that our initial feature model treats all features as independent entities. However, objects from the same class are likely to produce co-occurring features within confined regions, i.e. their bounding boxes. Over multiple bounding boxes containing instances from the same object class, indicative features are likely to be observed together. We present an algorithm which, based on the HOG features, aims at finding bounding boxes within the positive images which share such feature co-occurrences.

In the following section, we explain how we therefore represent bounding boxes by Bag-of-Visual-Words (BOW) histograms which are histograms over the occurrence frequencies of clustered local visual features.

Afterwards, we formulate the task of finding bounding boxes based on the BOW histograms as a latent structured learning problem in Sections 6.2 and 6.3. We then explain the algorithm we use for solving this problem in Section 6.4 whereas the latent structured problem formulation is based on [26] and [29].

Finally, in Section 6.5 we explain how we implement the CCCP algorithm [27] for solving the learning problem and finding improved bounding box estimations.

6.1 Feature representation

From our initial statistical model, we obtain a set of estimated rectangles $\hat{R}_I$ per image. We now want to describe each rectangle from $\hat{R}_I$ by a fixed-size feature vector based on our

HOG features. Since our estimations may be inaccurate and include relatively large background areas, we cannot expect to find similar HOG cells at similar relative locations within different rectangles. Therefore we choose a Bag-of-Visual-Words histogram representation which discards the locations of the local features and only counts their occurrence frequencies. For each rectangle $\hat{r}_{i,I}$ we thus obtain a BOW histogram with 10,000 occurrence frequencies.

Since HOG features are computed on a fixed grid we must find a grid rectangle which represents $\hat{r}_{i,I}$. For this purpose, we first define a set of reasonable aspect ratios on our grid which range from 15×7 to 7×15 HOG cells. We thus postulate that all possible objects have an aspect ratio of 0.5 in the extreme case and can be found with roughly 100 HOG cells on some scale.

Then, we search all image scales for the one rectangle which has one of these aspect ratios and at the same time the largest overlap with $\hat{r}_{i,I}$. Note that the number of scales considered depends on image size and on the number of scales which are required for extracting feature vectors for all examples. Counting the occurrence frequencies of the visual words within the respective grid rectangle then yields our BOW histogram. Each initial rectangle thus yields one training example $x_{i,I}$ for the learning problem below.

For the sake of readability, we omit the image index I for our rectangle estimations $\hat{r}_i$ and subsequently for training examples x_i for the remainder of this work. Implicitly, multiple rectangles or examples may hence stem from the same image in the following.

6.2 Structured learning problem

We first restate the structured learning problem as defined in [25], since the latent optimization problem explained below is based on it. A structured learning problem is characterized by a multi-dimensional label space Y. Therefore, a decision function $f_{\mathbf{w}}(x) : \mathbb{R}^d \rightarrow \mathbb{R}^k$ for a structured problem maps a d-dimensional feature vector to a k-dimensional output space. As a consequence, our feature representation $\Psi(x, y)$ for example x may also depend on the label y, i.e. x may be an image and y may be a rectangle determining the region for which a BOW histogram is extracted. Learning a linear maximum margin decision function $f_{\mathbf{w}}(x) = \langle \mathbf{w}, \Psi(x, y) \rangle$ involves solving a constrained optimization problem for model vector $\mathbf{w}$:

$$\mathbf{w}^* = \underset{\mathbf{w}}{argmin} \left(\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n+m} \xi_i \right) \tag{7}$$

$$\begin{aligned} s.t. \quad & \forall i \in \{1, ..., n+m\} : \forall \hat{y} \in Y \setminus y_i : (\langle \mathbf{w}, \Psi(x_i, y_i) - \Psi(x_i, \hat{y}) \rangle) \\ & \geq \Delta(\hat{y}, y_i) - \xi_i \end{aligned} \tag{8}$$

The constraints require that, under model vector $\mathbf{w}$, for each prediction $\hat{y}$ the difference between the original feature representation $\Psi(x_i, y_i)$ of example x_i and the predicted representation $\Psi(x_i, \hat{y})$ must have a score which exceeds the value of a loss function $\Delta(\hat{y}, y_i)$ quantifying the error committed by hypothesis $\hat{y}$. In other words, the score required for the difference between the correct feature representation and the predicted representation depends on "how incorrect" the prediction is. As for standard soft-margin classifiers, slack variables ξ_i soften the constraints, i.e. allow violations. Maximizing the margin while minimizing over the sum of the slacks as stated in (7) yields an optimum solution with minimum constraint violations.

6.3 Latent variables

A latent learning problem is a problem which involves unobserved properties which are modeled by latent variables. In our case, the latent variables h_i represent the unknown actual best bounding box for object instance i, i.e. in fact each latent variable is five-dimensional, since it models the $x-$ and $y-$ position as well as the width, height, and HOG grid scale. Object instances are created by the initial model and come with an initial estimation $\hat{r}_i$ for the latent variable h_i. Note that h_i also implicitly contains a scale on the multi-scale HOG grid which is initialized as explained above. We assume that all object instances are positive examples while all rectangles in negative images are negative examples, thus we assign a label $y_i \in \{-1, 1\}$ to each example. Overall, our examples have thus the form (x_i, y_i, h_i) whereas x_i denotes the grid of visual HOG words of image i.

It should be mentioned in this context that in our implementation, the label y_i is a binary label and our problem is thus only a structured problem in terms of the latent variable space. Adapting the terminology of [29] we refer to our problem as a latent structured problem. From structured learning we borrow the notation of $\Psi(x_i, y_i, h_i)$ for denoting the feature vector we obtain if we extract a BOW histogram from rectangle h_i on HOG grid x_i. Also, we treat our problem like a structured problem during the model vector training phase of the CCCP algorithm as explained below.

Now our problem is to find the optimal setting for the latent variables over all positive examples x_i. Inspired by [29], we use the CCCP algorithm proposed by [27] which iteratively trains an SVM model based on the current setting of latent variables and then applies this model to the training images in order to estimate new values for latent variables.

Overall, we need to learn a linear decision function which predicts the most likely label of an example i along with the optimal rectangle location:

$$f_{\mathbf{w}}(x_i) = \underset{(y_i,h_i)\in Y\times H}{argmax} \langle \mathbf{w}, \Psi(x_i, y_i, h_i)\rangle. \tag{9}$$

If we extend the structured optimization problem of (7) for latent variables as shown in [26], we obtain the following optimization problem:

$$\underset{\mathbf{w}}{min}\,(f(\mathbf{w}) - g(\mathbf{w})) \tag{10}$$

where

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n+m} \underset{(\hat{y},\hat{h})\in Y\times H}{max} (\langle \mathbf{w}, \Psi(x_i, \hat{y}, \hat{h})\rangle + \Delta(\hat{y}, \hat{h}, y_i)) \tag{11}$$

and

$$g(\mathbf{w}) = C\sum_{i=1}^{n+m} \underset{h\in H}{max} \langle \mathbf{w}, \Psi(x_i, y_i, h)\rangle. \tag{12}$$

This optimization problem includes the maximization of the margin and minimization of slacks in $f(\mathbf{w})$ which implicitly includes the constraints of (8) with added latent variables. Also, the formulation incorporates the maximization over the latent variables in $g(\mathbf{w})$. Also following [29] and [26], we solve this non-convex optimization problem by applying the CCCP algorithm as explained in the following section.

6.4 Training algorithm

Our training algorithm is an instance of the Convex-Concave Procedure (CCCP) by Yuille and Rangarajan [27]. Algorithm 1 provides a brief description of the CCCP algorithm in our scenario.

In line 1 our latent variables are initialized. Note that this applies only to positive examples. For negative examples, we simply begin with pseudo-random rectangles which roughly describe the full negative images. In later iterations, the negative examples are mined as explained below.

Line 2 contains the main loop over the iterative process of training a linear model and inferring latent variables. Analogous to [26] we begin each iteration by constructing a linear upper bound to the objective function $f(\mathbf{w}) - g(\mathbf{w})$ of (10) which we then minimize instead of the non-convex $f(\mathbf{w}) - g(\mathbf{w})$. The linear upper bound is given in line 4 and based on the hyperplane constructed in line 3 which constitutes a linearization of the objective function.

Algorithm 1: CCCP training algorithm

Input
n positive and negative examples $(x_1, y_1, h_i), ..., (x_n, y_n, h_i)$;
Initial rectangle estimations $\hat{r}_i$ for positive examples

1 $\forall i : h_i^* = h_i = \hat{r}_i$; $\mathbf{w}_0 = \mathbf{0}$; $t = 0$;
2 **repeat**
  `// Construct upper-bound hyperplane for objective function`
3 $\mathbf{v}_t = \sum_{i=1}^{n+m} \Psi(x_i, y_i, h_i^*)$;
  `// Solve optimization for linear lower bound of objective function`
4 $\mathbf{w}_{t+1} = \underset{\mathbf{w}}{argmin}[f(\mathbf{w}) + \langle \mathbf{w}, \mathbf{v}_t \rangle]$;
  `// Infer latent variables` h_i^* `for each positive example` $x_i \in X$
5 $h_i^* = \underset{h \in H_i}{argmax} \langle \mathbf{w}_{t+1}, \Psi(x_i, y_i, h) \rangle$;
6 $t = t + 1$;
until $(f(\mathbf{w}_{t+1}) - g(\mathbf{w}_{t+1})) - (f(\mathbf{w}_t) - g(\mathbf{w}_t)) < \epsilon$;

Output
Latent variable h_i^* for each positive example

Minimizing this upper bound yields a model vector $\mathbf{w}$ which we then use in line 5 for inferring new values of the latent variables, i.e. the bounding box rectangles for the positive examples. Both the optimization step in line 4 and the inference step in line 5 require some implementation decisions which we elaborate in the next section.

6.5 Implementation decisions

For our CCCP algorithm we need a training strategy for the optimization step in line 4 and an inference method in line 5. In this section we explain our implementations of both steps in detail.

6.5.1 Model training

Our objective function is $f(\mathbf{w}) - g(\mathbf{w})$ whereas minimizing $f(\mathbf{w})$ corresponds to finding a maximum decision margin and $-g(\mathbf{w})$ is introduced for finding the best setting for latent variables. Since we assume fixed latent variables for this step, we can at this point interpret

the problem as a (non-latent) structured constrained learning problem. For this purpose, the latent variable h_i now takes the role of the structured label in a non-latent structured problem, i.e. we use h_i as the current true bounding box and $\hat{h}$ denotes a bounding box prediction.

This problem can then be solved using the Cutting Plane algorithm by [10]. The Cutting Plane algorithm iteratively trains an SVM model by building sets of constraints and then solving the resulting constrained optimization problem for a model vector $\mathbf{w}$. The constraints of our problem have the form

$$\forall \hat{y}, \hat{h} \in Y \times H : \langle \mathbf{w}, \delta\Psi \rangle \geq \Delta(\hat{y}, \hat{h}, y_i) - \xi_i \tag{13}$$

where

$$\delta\Psi = \Psi(x_i, y_i, h_i) - \Psi(x_i, \hat{y}, \hat{h}) \tag{14}$$

and $\Delta(\hat{y}, \hat{h}, y_i)$ is the loss function quantifying the error of estimated label $\hat{y}$ and rectangle $\hat{h}$. The loss function may not depend on h_i which is a prerequisite for deriving the objective function of the latent structured problem as elaborated in [26]. Following [29], we simply use a straightforward binary loss function, i.e. $\Delta(\hat{y}, \hat{h}, y_i) = d(y_i \neq \hat{y})$ where $d(A) = 1$ if A is true and 0 otherwise. We discuss why this is a reasonable choice for our problem below.

A rectangle estimation $\hat{h}$ is only meaningful for a positive example or, in other words, the correct prediction for a negative example would be an empty rectangle (denoted by $\emptyset$) and hence an empty feature vector. As a consequence, we define the representation of an empty prediction as $\Psi(x_i, \hat{y} = -1, \hat{h} = \emptyset) = \mathbf{0}$ where $\mathbf{0}$ is a zero-vector.

For building the set of constraints for the optimization problem, we now search each training example for the most violated constraint, i.e. the most useful constraint for the current iteration. Positive examples may thereby either be predicted correctly or incorrectly, whereas an incorrect prediction is made if the empty prediction's score (i.e. 0) is larger than or equal to the score of the actual positive example's feature vector (which in this case would be ≤ 0). We do not consider rectangles which are different from the current estimation h_i in positive images, since quantifying the error for partially correct rectangles is not applicable in our scenario as also explained below.

For negative images we know that each predicted rectangle is fully incorrect. As a consequence, we search the highest scoring rectangle $\hat{h}$ in each negative image for building a most violated constraint. Again, the highest scoring prediction may also be empty which is the only correct prediction for a negative example.

Training examples which we already predict correctly are not useful for training our model which is reflected by the fact that such examples do not yield any non-trivial constraint. This can be verified by substituting the respective predictions and loss value of $\Delta(\hat{y} = y_i, \hat{y}, y_i) = 0$ in (13). For a correct positive or negative prediction, $\delta\Psi$ becomes $\mathbf{0}$ and altogether the constraint becomes trivial (i.e. inactive), since the current model $\mathbf{w}$ already fulfills it.

Positive training examples which are predicted incorrectly by $(\hat{y} = -1, \hat{h} = \emptyset)$, however, yield constraints of the form

$$\langle \mathbf{w}, \Psi(x_i, y_i, h_i) \rangle \geq 1 - \xi_i \tag{15}$$

whereas incorrectly predicted negative examples with $(\hat{y} = 1, \hat{h} \neq \emptyset)$ provide constraints of the form

$$-\langle \mathbf{w}, \Psi(x_i, y_i, \hat{y}) \rangle \geq 1 - \xi_i. \tag{16}$$

since $\delta\Psi$ becomes the (positive or negative) feature representation $\Psi(x_i, y_i, h_i)$ of example i. Thus, if we predict a rectangle $\hat{h}$ in a negative example other than the empty rectangle, it will provide a constraint which requires that the respective feature vector obtains a score $\leq -1 + \xi_i$ under model vector $\mathbf{w}$. Note that this is equivalent to a constraint from a standard (non-structured) constrained linear SVM optimization problem.

After building the set of constraints, we solve the actual optimization problem using SVMLight [9]. Searching the most violated constraint in negative images involves evaluating a linear model for a large number of rectangles, so we build integral images over visual-word-contributions as suggested in [12] which reduces evaluating a rectangle to four lookups.

6.5.2 Inference of latent variables

In line 5 of our algorithm, we update the latent variables of all positive examples. This basically corresponds to finding the highest scoring rectangle within each positive image under the current model vector. However, we reduce the search space in two ways.

First, we require that the new rectangle has some minimum overlap (0.1) with the initial rectangle, because otherwise, all instances would converge to the globally optimal rectangle of the respective image instantly. Note that this does not prevent multiple initial estimations form converging to the same object if the initial rectangles are close to each other. This is actually desired behavior because it enables us to reject multiple estimations covering the same object instance during non-maximum suppression. Second, we limit the number of valid aspect ratios, since using a linear model based on a BOW histogram representation which omits all spatial information tends to converge to small rectangles and is not constrained against other unreasonable rectangle shapes.

For this purpose we identify the one aspect ratio which best fits the majority of positive examples from the above mentioned set of ratios between 15×7 and 7×15 in a two-stage process: First we determine if the majority has horizontal, vertical or square aspect ratios. Then we select the one aspect ratio within the respective group which explains the most examples. This aspect ratio is used as a reference and is updated for each CCCP iteration. We then allow all aspect ratios which deviate at most 2 HOG cells in each dimension from our reference ratio whereas we forbid shrinking with regards to included number of cells in order to prevent the aforementioned converging to very small rectangles. Note that for searching the most violated constraints in negative images we use the same aspect ratios, since we want negative examples with consistent rectangle shapes.

After the final iteration of the CCCP algorithm, we post-process the resulting bounding boxes. First, we perform the inference on double-resolution versions of the training images which enables us to find very small object instances more accurately. We only do this for detections on the lowest and second to lowest scales. Also, for such small detections, we further shrink the borders towards positive pixels from the initial model in order to better detect even smaller objects. As a second step, we perform non-maximum suppression by removing all bounding boxes which overlap more than 2/3 with a higher-scoring rectangle. Note that this requires a scoring function which is provided by the CCCP algorithm and which is not available for the initial model. Finally, boxes which do not deviate more than one HOG cell in any direction from the respective initial estimation are replaced by the latter. The reason for this step is that in this case the initial model and the CCCP algorithm are likely to describe the same image region whereas the initial

rectangle which lives on the pixel-grid instead of the coarser HOG-grid is potentially more accurate.

7 Evaluation

For evaluating how accurate our bounding box estimations are we plot their overlaps with ground truth annotations. Plotting the overlap for each object instance present in the positive images yields an overlap-recall (OR) curve. More precisely, we compute the overlap for each object instance in the dataset and then plot the resulting overlap values in descending order. Since OR curves do not show false positive detections and can be improved by simply increasing the number of hypotheses (estimated rectangles) per image, we also state the average absolute number of false positive detections in the legends of the plots. Note that this number is not to be confused with a relative false positive rate which may -depending on the dataset- be significantly smaller since we often have a huge number of possible rectangles which do not overlap with an object instance. A false detection is defined as a detection which does not overlap with any instance of the desired object. A detection which only produces overlap with an object instance already covered by another detection with better overlap also counts as a false detection as is common in such evaluations (i.e. in [7]).

We use three publicly available datasets for our experiments for which our assumptions stated in Section 3 hold to a certain degree. Since our datasets have 32, 17, and 10 classes, respectively, we cannot show the results for each individual class. We thus aggregate the results over all classes into a single plot for each dataset.

In each plot, the results of the CCCP algorithm are shown by a green OR curve. As a baseline we show the result of our initial model labeled as "color AND hog multi" by a red curve. For comparison, we also use the Objectness measure as a second baseline in each plot by a purple dashed curve. The Objectness measure and our motivation for choosing it as a baseline are discussed in Section 7.4.

Even though we propose an offline method and a complexity analysis is beyond the scope of this work, we still point out that the algorithm requires a reasonable amount of time. On a 16-core workstation one class is finished after roughly 20 to 30 minutes depending on the number of images used.

7.1 FlickrLogos-32

Our first dataset is FlickrLogos-32 [20] which consists of 32 classes of brand logos, each with 70 images. The dataset also contains a negative set of 6,000 images which does not contain any of the 32 brand logos and thus serves as our background set. For the initial HOG model we use the feature representation of the full negative images and add 5 random bounding boxes per negative image for a total of 36,000 negative examples. The authors of the dataset provide ground truth bounding boxes for evaluation.

The OR curve in Fig. 4 shows our results on this dataset. Overall, the CCCP algorithm yields better annotations with regard to overlap with manual annotations then the initial model. Also, the CCCP algorithm considerably reduces the number of false detections.

A few qualitative examples are shown in Fig. 5. The green rectangles are the bounding boxes found by the CCCP algorithm while the red rectangles are found by the initial

model. The white rectangles are manual ground truth annotations. The first three examples show images where the CCCP result is better than the initial model. We also show examples for two error types: In the second example, a missed instance is shown. The third example shows a false detection indicated by the dark green rectangle. In the fourth and fifth example, the initial model yields a better bounding box due to a bad global aspect ratio and a partial detection, respectively. We want to point out that in some cases, the rectangle we find does actually cover an indicative part of the respective object class while producing a poor overlap with the ground truth. In the fifth example of Fig. 5, for instance, the overlap of the CCCP result is considerably lower than for the initial model, which almost exactly covers the ground truth (which is hence hardly visible). Still, the rectangle found by the CCCP algorithm covers the "main" part of the logo which implicitly is the goal of our approach. The final example in Fig. 5 shows an image where the CCCP algorithm fails to converge towards the actual object due to background clutter and a poor initial estimation (which is the full image in this case).

For the FlickrLogos-32 dataset, we also show the OR curves for 6 selected classes: "DHL", "Aldi", "Coca Cola", "Pepsi", "Shell", and "Esso" in Fig. 6. We choose the same classes as in [17] which we feel are fairly representative for the whole dataset. For all classes except "Shell" the CCCP algorithm improves the results.

The "Shell" class is an example for the minority of classes where the CCCP algorithm fails to improve the results. The initial model works exceptionally well on the "Shell" class while the CCCP algorithm is less flexible due to its global reference aspect ratio which is not estimated very accurately for this class as the lower left example of Fig. 5 indicates.

The number of false detections is reduced by the CCCP algorithm for all of our selected classes except "Pepsi". This means that for "Pepsi" a few instances converge from rectangles overlapping with an object instance towards regions which are completely in the background. The reason is that for Pepsi, the CCCP algorithm moves rectangles from the annotated circular logo towards the non-annotated writing of the word "Pepsi" which is also present in most positive images.

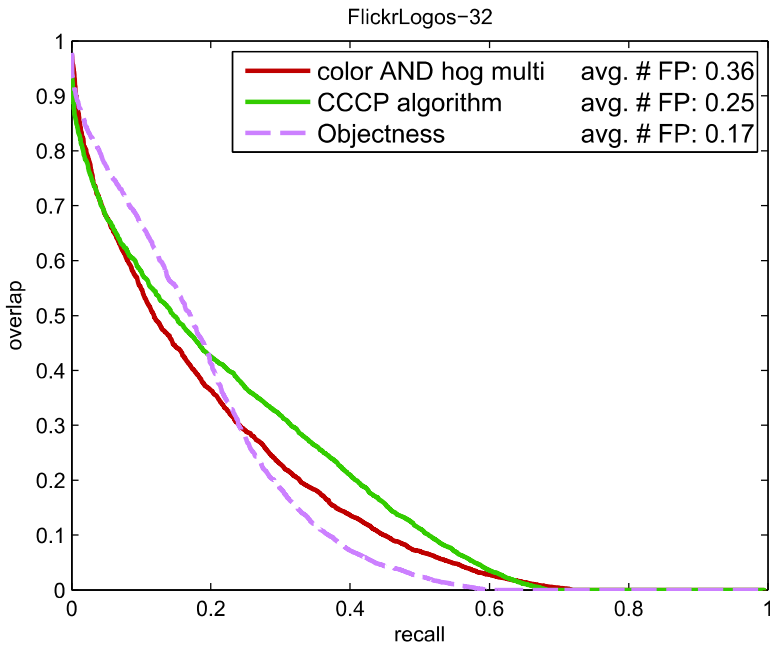

Fig. 4 Overlap-recall curve for FlickrLogos-32 dataset

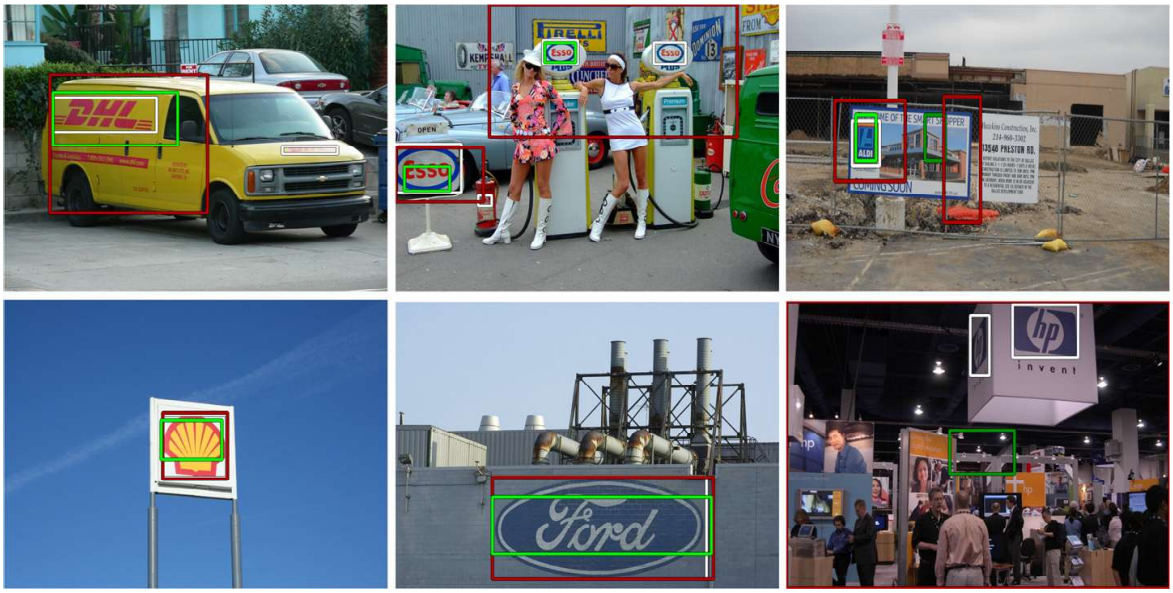

Fig. 5 Qualitative example results for FlickrLogos-32: *Red box* is returned by the initial model, *green box* is the result of CCCP, white is the manually annotated ground truth

7.2 Oxford 17 flowers

Even though the task of annotating flowers may arguably not be a highly relevant one, the Oxford 17 Flowers dataset [14] is still an interesting dataset for our approach. The reason is that Flowers usually have strong discriminative color features while gradient-based features are less suitable. The dataset consists of 17 flower classes and provides 80 images for each class. For a subset of 848 images pixel-annotations are provided from which we infer ground truth bounding boxes by creating rectangles around connected components. Note that this process may result in a few incorrect annotations for overlapping instances, even though we have re-annotated the majority of these cases. Also note that the number of annotations per class varies and one class has no annotations at all.

The OR curves are shown in Fig. 7. For the Flowers dataset, the CCCP algorithm does not improve the bounding boxes beyond the initial model. The reason is that the initial model uses color features which are very suitable for flowers. However, the CCCP algorithm again reduces the number of false detections.

A few examples are shown in Fig. 8. In the first example, the CCCP algorithm slightly improves the bounding box. In the remaining examples, the initial model yields better estimations. The third example shows a situation where the CCCP algorithm loses one object instance since both object instances are close enough for iteratively converging towards two close rectangles whereas one does not survive non-maximum suppression.

7.3 3D object categories

Our final dataset is the 3D Object Categories dataset [21] which consists of images from 10 different object classes such as “car”, “bicycle”, or “stapler”. This dataset is arguably the opposite of the Flowers dataset with regards to our features, since the object classes are usually not characterized by strong discriminative colors. However, we expect stronger gradient features than for flowers.

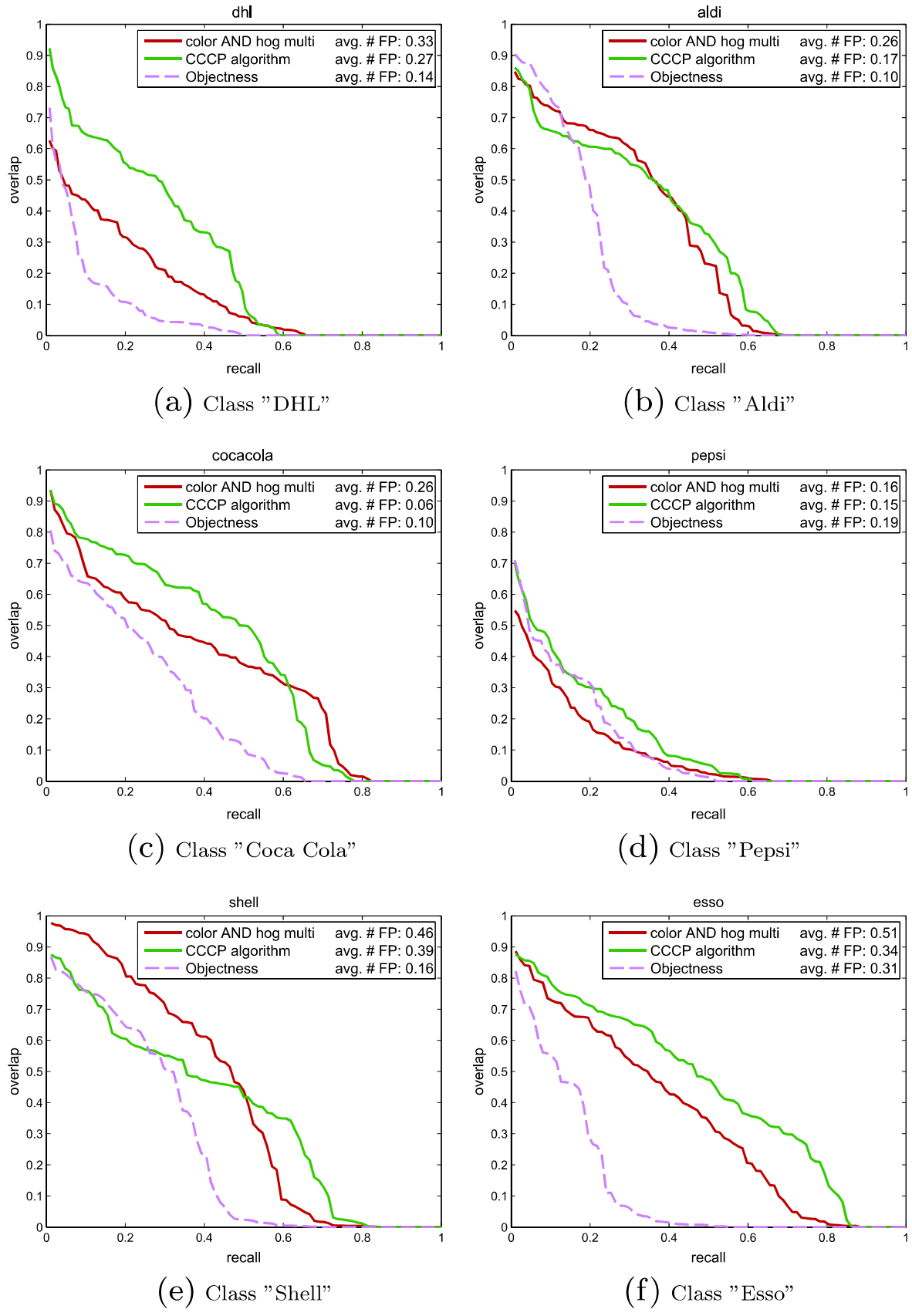

Fig. 6 OR curves for six logo classes from FlickrLogos-32

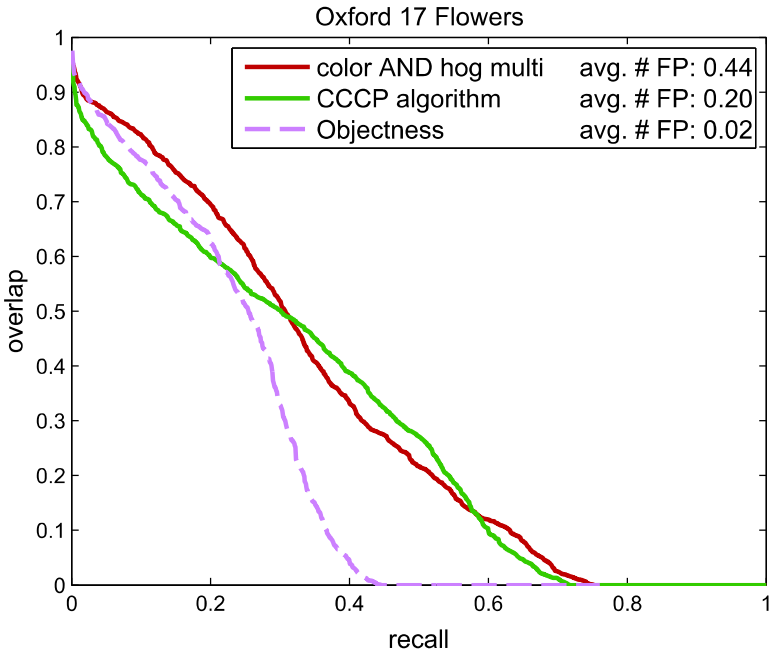

Fig. 7 Overlap-recall curve for Oxford 17 Flowers dataset

Each class of the 3D Object Categories dataset consists of 10 different object instances which are photographed against different backgrounds from different pre-defined points of view. Since some objects virtually become visually different classes from different perspectives, we only use 15 different versions of each instance which still vary considerably with regards to point of view. For some objects not all viewpoints are available, so we obtain between 115 and 150 per class. Note that for this dataset we do not search double-resolution images as post-processing due to the low image quality.

Figure 9 shows the OR curves. The CCCP algorithm slightly outperforms the statistical model and reduces the number of false detections, since the HOG features are relatively suitable for the dataset while the color features do not provide much advantage as for the flower classes. Recall that the OR curve shows aggregated results for all 10 classes. In fact, the CCCP algorithm yields better results on 7 of the 10 classes.

In Fig. 10 we again show three example results. The rightmost example in Fig. 10 shows an image where the CCCP algorithm removes one false detection of the initial model during non-maximum suppression.

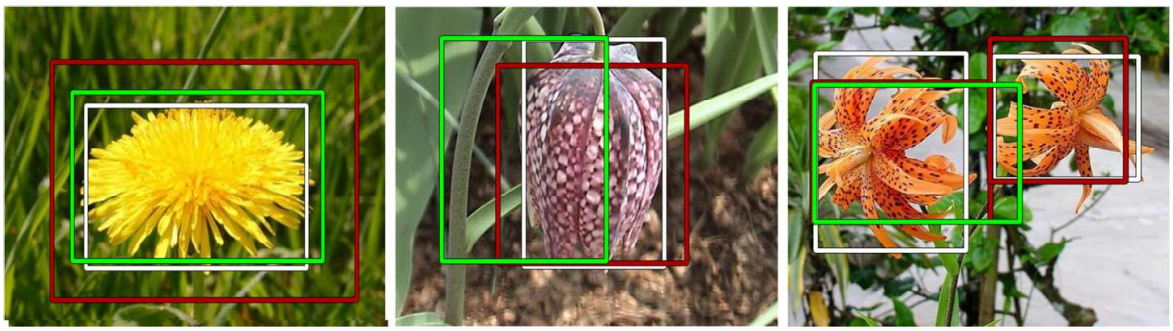

Fig. 8 Qualitative example results for Oxford 17 Flowers: *Red box* is returned by the initial model, *green box* is the result of CCCP, white is the manually annotated ground truth

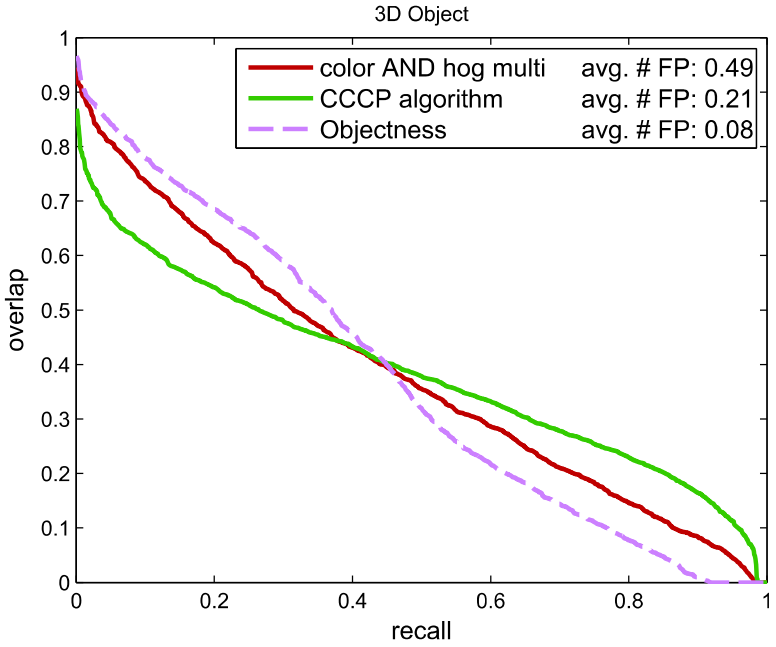

Fig. 9 Overlap-recall curve for 3D Object Categories dataset

7.4 Comparison to objectness measure

For comparison, we use the approach by Alexe et al. [1, 2], which finds bounding boxes which are likely to contain an object by assigning an "Objectness" score to bounding boxes based on a five-feature model which is trained on manually annotated objects from the VOC challenge 2007 dataset [7]. For using it in the context of automatic annotation, we simply retrieve the most likely object location according to the Objectness measure for each positive image. We do not use multiple hypotheses in order to keep the results reasonable with regards to false detections and comparable to the CCCP approach as discussed below.

We want to emphasize that the Objectness measure is not designed for our definition of the problem of automatic annotation. In particular, the idea of the Objectness measure is to find *any* object in an individual image and does not aim at finding instances of the *same* object class across a set of images. This has two important consequences: First, our evaluation does not make a statement about the performance of the Objectness measure for its intended use but only in the context of our problem. Second, if we use the Objectness

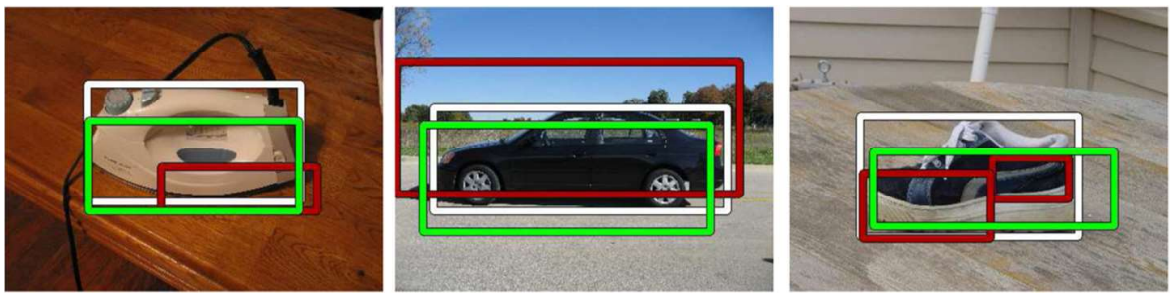

Fig. 10 Qualitative example results for 3D Object Categories: *Red box* is returned by the initial model, *green box* is the result of CCCP, white is the manually annotated ground truth

measure in our context, we implicitly change our assumptions, since if only one object is present in the image, the Objectness measure often finds it with almost perfect accuracy. This, however, is a special case where the positive images do not show any object besides the wanted object.

We think that it is interesting to compare the Objectness measure to our approach because the problem of finding any object is very similar to our problem. By evaluating the Obejctness measure we implicitly show under which circumstances our problem is different from finding any object and, as a consequence, in which cases we actually need an approach which models common feature apperances over all images of the positive set. In other words we show that using our approach can be justified as it actually determines the locations of the desired object instances and not merely regions of interest.

We show in each OR curve the result of the Objectness measure whereas we simply accept the highest scoring rectangle as estimation of the wanted object's location. Since we only have one detection per image for the Objectness measure, it obviously has a lower false detection rate. Our approach combines evidence from all positive images in the learning algorithm, and thus obtains better OR curves for FlickrLogos-32 and Oxford 17 Flowers, since more object instances are found.

For the 3D Object Categories dataset, the Objectness measure yields an area under the curve which is only slightly smaller than for our method with a lower number of false detections. The reason for this result is that the 3DObjects dataset does in most images not show any distracting objects beyond background clutter, so the rectangle with the highest Objectness score is often the wanted object. As mentioned above, this contradicts our problem definition, because we do not assume that there are no other objects present in the positive images.

Interestingly, for all classes the Objectness curves begin at very high overlaps. The reason is that if the Objectness method detects the wanted object it is relatively likely to also determine its exact bounds.

We also briefly discuss increasing the number of hypotheses, i.e. top scoring bounding boxes returned per image, for the Objectness measure. Note that the Objectness method performs non-maximum suppression for detections overlapping by more than 50%, i.e. rectangles are found which are fairly different and it is comparable to our approach. If we allow 2 hypotheses, the area under the curve is roughly on par with the result of the CCCP algorithm for Logos and Flowers and slightly better for the 3D Objects Categories dataset. However, we then obtain almost twice the amount of false detections of the CCCP algorithm and overall we obtain roughly as many false detections as true detections. For automatic annotation, this is not considered a reasonable result. With 3 or more hypotheses per image, the false detections clearly outweigh the correct ones while, obviously, the areas under the curves keep increasing.

7.5 Retrieval with automatic annotations

The overlap-recall statistics reveal how accurate our bounding boxes are compared to ground truth annotations. However, these statistics do not necessarily indicate the usefulness of the automatic bounding boxes for an actual application for two reasons: First, the overlap measure is relatively strict and the manual annotations are not always flawless. In particular, if objects are slightly rotated or their shape is naturally non-rectangular, the manual annotations will also include relatively large background areas. Second, the manual

annotations do not explicitly yield the most discriminative portions of the respective objects (see for instance the fifth example image (the "Ford" logo) in Fig. 5). In practice, however, many tasks only require an indicative subset of features from a given object class, as for instance discussed in [15]. One such task is image retrieval in the popular query-by-example variant. Query-by-example image retrieval denotes the task of finding images in a large database which show the same object (scene, concept, or similar) as a given query image. Clearly, this requires only matching a small subset of features from the object class at hand between query image and database images.

Our experimental setup follows [19] on the FlickrLogos-32 dataset. First we create an image database which contains both images from a given logo class and non-logo images whereas we use the training images as suggested by the authors of the FlickrLogos-32 dataset. For our database, an inverted index is created based on RootSIFT Bag-of-Words with tf-idf weighting and burstiness measure as described in [19].

In order to evaluate the quality of the retrieved image lists, we analyze the top k retrieval results, i.e. the k images which were returned with the highest similarity values with regards to our query image. We then retrieve the top-k most similar images from our database for each test image of the respective logo class. For these k images we perform straightforward $k$nn classification, i.e. a test image is counted as a true positive classification if the majority of the k most similar images belong to the same logo class. We then compute the average recall over all 32 classes. In this context "recall" is the number of query images for which the majority of k nearest nieghbors has the same class as the query image. Note that false positive matches reduce the recall in this scenario since they directly influence the $k$nn results, so this recall value also reflects false positive detections.

Four experiments are conducted. First, we use the full images for retrieval as a baseline, i.e. the RootSIFT features are extracted on the full positive images. Then, we use the bounding boxes found by our method for the training images, i.e. when building the inverted index, we ignore all features which do not fall into one of our bounding boxes. For comparison we also determine the results for Objectness-based bounding boxes. Finally, we also perform the retrieval exploiting manual annotations on pixel-level. Note that the latter is the theoretical optimum for object annotations. Therefore the respective results are the best possible results with regards to different types of annotations. The average recall values for all four experiments are shown in Table 1.

As expected, the manual annotations on pixel-level yield the best results while the Objectness-based boxes slightly improve the retrieval result compared to full-image retrieval. Our approach ("our boxes" in Table 1) improves the search and comes close to the manual annotations for small values of k, i.e. informally speaking the top nearest neighbors are improved. For instance at $k = 1$ and $k = 2$, the performance is only 0.01 below the

Table 1 Average recall values of $k$nn classification on top k retrieval results on FlickrLogos-32

k	1	3	5	7	9
Full images	0.83	0.81	0.79	0.78	0.77
Objectness	0.84	0.81	0.80	0.78	0.78
Our boxes	0.86	0.83	0.82	0.81	0.80
Human annotations	0.87	0.85	0.85	0.84	0.84

respective theoretical maximum which exploits manual pixel-wise annotations. We therefore conclude that our automatic annotations in fact remove irrelevant features and preserve the most important indicative features.

8 Discussion

In this section we discuss our methods in the context of the automatic annotation problem.

As mentioned in the beginning of this work, we only want to use global image labels and no further information. Thus, it seems reasonable that our initial statistical model directly works on statistics inferred from global image labels. Obviously, the discriminative power of this model is limited, but it is a straightforward approach which in our experiments for most classes finds indicative features if such features are present. Also, the approach can be easily extended to further feature types. However, correctly determined positive features may still lead to highly incorrect bounding boxes as can be seen in the upper row of images of Fig. 5 where the initial model considerably "over detects" the respective object instances.

Since the actual regions of interest are unknown and we do not want to introduce any further knowledge, we turn to a latent learning approach. The CCCP algorithm is hence applied as it solves latent learning problems. By implementing the unknown bounding box locations as latent variables, the CCCP algorithm can be used to find boxes based on a model for consistent feature co-occurrences (as opposed to simple occurrences of the initial model). Even though the CCCP algorithm is usually used for training a discriminative model, it still aims at finding the best settings for the latent variables on the positive training set which is what we are interested in.

In this context it is worth mentioning that we use a binary loss function for the CCCP algorithm instead of an overlap-based loss function as for instance proposed in [3]. An overlap-based loss function assigns an error < 1.0 to predictions on positive images which have an overlap of > 0 with the actual ground truth rectangle. It also allows searching for the most violated constraint in positive images in terms of determining the highest scoring rectangle. Since we, however, do not use ground truth rectangles and only have the often inaccurately estimated rectangles from our initial model (or the respective previous iteration of the CCCP algorithm), it is not reasonable to use a loss function which assigns different error values to different incorrect predictions. In many cases multiple predictions exist which have no overlap with the actual object instance but different overlaps with the current estimated rectangle. It is obvious that assigning different loss values to these predictions would be misleading our algorithm. Therefore we implicitly define the current estimation as the only correct prediction which is then updated in each iteration.

Another important issue are the results of our approach. It is important to emphasize that we do not claim to have solved the problem of automatic object annotations and there is obviously room for improvement as suggested by our OR-curves. However, as we have already addressed in the evaluation section, our results seem promising especially under the assumption that we do not necessarily want to find bounding boxes which are identical to subjective manual annotations. In many cases, it is already useful to simply remove background areas while in other cases, finding a discriminative part of an object is desirable, which is also indicated by the result of our retrieval experiment. Note that this is arguably an issue of the evaluation part in our problem definition which does not question manual bounding box annotations.

9 Summary and conclusion

In this work we have discussed the problem of automatic object annotation. We have presented an approach to this problem which is based on a statistical feature model which we have implemented for two different features. We then have explained an enhancement of this approach by a latent structured learning algorithm.

Our experimental evaluation has shown that we find the location of the wanted object in many cases to a certain degree on the majority of classes. We have also compared our approach to a method which determines the locations of any object within our positive images and thereby show that our discriminative models go beyond detecting salient regions.

A considerable number of object instances are, however, only found with relatively small overlap for some classes leaving room for improvement in future work. As indicated by the OR curves on individual classes in Fig. 6, the quality of results of our approach highly varies for different classes. For some classes, however, certain features are more suitable. For instance, for most flower classes colors are suitable while gradient-based features are not very distinctive in the context of our task.

Since we do not exploit any information except global image labels, many implementation decisions rely on empirically selected parameters. In fact, our experimental results can be significantly improved by adjusting parameters for each class individually which we think shows that the approach is promising but improvable. Also, additional features may improve our method or make it usable for more challenging datasets. For example, different local descriptors may capture further visual evidence which can be found among multiple instances of an object class.

As discussed above, our approach does not ultimately solve the problem of automatic object annotation. However, we consider our results promising for future work. Besides presenting our suggestions for tackling this problem, the intention of this work is also to discuss our view on this recent and relevant topic and to encourage other researchers to contribute theirs.

References

1. Alexe B, Deselaers T, Ferrari V (2010) What is an object? In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR '10
2. Alexe B, Deselaers T, Ferrari V (2012) Measuring the objectness of image windows. IEEE Trans Pattern Anal Mach Intell 34(11):2189–2202
3. Blaschko MB, Lampert CH (2008) Learning to localize objects with structured output regression. In: Proceedings of European conference on computer vision, ECCV '08, pp 2–15
4. Chen CY, Grauman K (2013) Watching unlabeled video helps learn new human actions from very few labeled snapshots. In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR '13
5. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR '05, vol 1, pp 886–893
6. Dietterich TG, Lathrop RH, Lozano-Prez T (1997) Solving the multiple instance problem with axis-parallel rectangles. Artif Intell 89(1-2):31–71
7. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2007) The PASCAL visual object classes challenge (VOC2007) results http://www.pascal-networkorg/challenges/VOC/voc2007/workshop/indexhtml
8. Felzenszwalb P, Girshick R, McAllester D, Ramanan D (2010) Object detection with discriminatively trained part-based models. IEEE Trans Pattern Anal Mach Intell 32(9):1627–1645

9. Joachims T (1999) Making large-scale svm learning practical. In: Advances in kernel methods - support vector learning. MIT Press, pp 169–184
10. Joachims T, Finley T, Yu CN (2009) Cutting-plane training of structural svms. Mach Learn 77(1):27–59
11. Jones MJ, Rehg JM (2002) Statistical color models with application to skin detection. Int J Comput Vis 46(1):81–96
12. Lampert C, Blaschko M, Hofmann T (2009) Efficient subwindow search: a branch and bound framework for object localization. IEEE Patt Anal Mach Intell 31(12):2129–2142
13. Maron O, Ratan AL Multiple-instance learning for natural scene classification. In: Proceedings of international conference on machine learning 1998, ICML '98
14. Nilsback ME, Zisserman A A visual vocabulary for flower classification In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR '06
15. Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2008) Lost in quantization: improving particular object retrieval in large scale image databases. In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR '08, pp 1–8
16. Ries CX, Lienhart R (2012) Deriving a discriminative color model for a given object class from weakly labeled training data. In: Proceedings of ACM international conference on multimedia retrieval, ICMR '12, pp 44:1–44:8
17. Ries CX, Richter F, Lienhart R (2013) Towards automatic object annotations from global image labels. In: Proceedings of ACM conference on international conference on multimedia retrieval, ICMR '13, pp 207–214
18. Ries CX (2014) Automatic object annotations from weakly labeled images. Dissertation, University of Augsburg
19. Romberg S, Lienhart R (2013) Bundle min-hashing. Int J Multimed Inf Retr 2(4):243–259
20. Romberg S, Pueyo LG, Lienhart R, van Zwol R (2011) Scalable logo recognition in real-world images. In: Proceedings of ACM international conference on multimedia retrieval, ICMR '11, pp 25:1–25:8
21. Savarese S, Fei-Fei L (2007) Generic object categorization, localization and pose estimation. In: Proceedings of IEEE international conference on computer vision, ICCV '07
22. Siva P, Russell C, Xiang T (2012) In defence of negative mining for annotating weakly labelled data. In: Proceedings of European conference on computer vision, ECCV '12, pp 594–608
23. Sivic J, Zisserman A (2003) Video google: a text retrieval approach to object matching in videos. In: Proceedings of IEEE international conference on computer vision, ICCV '03, vol 2, pp 1470–1477
24. Tang K, Rahul S, Jay Y, Li FF (2013) Discriminative segment annotation in weakly labeled video. In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR '13
25. Tsochantaridis I, Hofmann T, Joachims T, Altun Y (2004) Support vector machine learning for interdependent and structured output spaces. In: Proceedings of international conference on machine learning, ICML '04, pp 104–
26. Yu CNJ, Joachims T (2009) Learning structural svms with latent variables. In: Proceedings of international conference on machine learning, ICML '09, pp 1169–1176
27. Yuille AL, Rangarajan A (2003) The concave-convex procedure. Neural Comput 15(4):915–936
28. Zhang C, Platt JC, Viola PA (2005) Multiple instance boosting for object detection. In: Weiss Y, Schölkopf B, Platt J (eds) Advances in neural information processing systems, vol 18, pp 1417–1424
29. Zhu L, Chen Y, Yuille AL, Freeman WT (2010) Latent hierarchical structural learning for object detection. In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR '10, pp 1062–1069

Christian X. Ries acquired the Ph.D. degree at the Multimedia Computing Lab of the University of Augsburg in 2014. His research interests are in the areas of Computer Vision, Machine Learning, and Image Content Analysis.

Fabian Richter works as a PhD student at the Multimedia Computing and Computer Vision Lab, University of Augsburg, Germany. He received his diploma degree in Computer Science from the Uni- versity of Augsburg, in November 2009. His re- search interests include Machine Learning, Computer Vision and Document Analysis.

Rainer Lienhart is a full professor in the computer science department of the University of Augsburg. He received his Ph.D. in Computer Science from the University of Mannheim, Germany, in 1998, where he was a member of the Movie Content Analysis Project (MoCA). From August 1998 to July 2004 he was a Staff Researcher at Intel's Microprocessor Research Lab in Santa Clara, California, where he worked on transforming a network of heterogeneous, distributed computing platforms into an array of audio/video sensors and actuators capable of performing complex DSP tasks such as distributed beamforming, audio rendering, audio/visual tracking, and camera array processing. At the same time, he was also continuing his work on media mining, where he is well-known for his work in video content analysis with contributions in text detection/recognition, commercial detection, face detection, shot and scene detection, and automatic video abstraction. He as been a committee member of ACM Multimedia, IEEE International Conference on Multimedia Systems, IEEE International Conference on Multimedia Expo, SPIE Storage and Retrieval of Media Databases, IEEE Workshop on Content-Based Access of Image and Video libraries and the International Eurographics Workshop on Multimedia. He is a reviewer for IEEE Transactions on Pattern Recognition and Machine Learning, IEEE Transaction on Multimedia, Journal of Computer Vision and Image Understanding and ACM Multimedia Systems Journal. Dr. Lienhart has published over 50 papers in major conferences and journals and filed 20+ patents.