Towards Automatic Object Annotations from Global Image Labels

Christian X. Ries
Augsburg University
Universitätsstr. 6a
86150 Augsburg
ries@informatik.uni-augsburg.de

Fabian Richter
Augsburg University
Universitätsstr. 6a
86150 Augsburg
richter@informatik.uni-augsburg.de

Rainer Lienhart
Augsburg University
Universitätsstr. 6a
86150 Augsburg
lienhart@informatik.uni-augsburg.de

ABSTRACT

We present an approach for automatically devising object annotations in images. Thus, given a set of images which are known to contain a common object, our goal is to find a bounding box for each image which tightly encloses the object. In contrast to regular object detection, we do not assume any previous manual annotations except for binary global image labels. We first use a discriminative color model for initializing our algorithm by very coarse bounding box estimations. We then narrow down these boxes using visual words computed from HOG features. Finally, we apply an iterative algorithm which trains a SVM model based on bag-of-visual-words histograms. During each iteration, the model is used to find better bounding boxes which can be done efficiently by branch and bound. The new bounding boxes are then used to retrain the model. We evaluate our approach for several different classes of publicly available datasets and show that we obtain promising results.

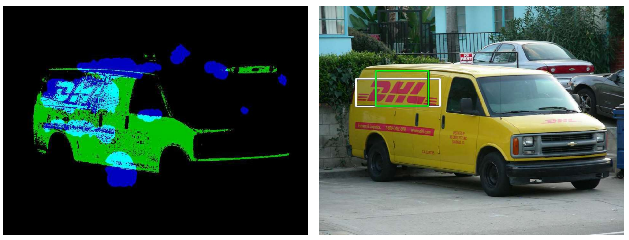


Figure 1: A sample result for an image from the FlickrLogos-32 dataset: The left image shows the result of applying a mined color model (dark green area), a HOG model (dark blue area), and the combination of both (cyan area). The right image shows the final result of our algorithm (green box, white box is the ground truth annotation).

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

automatic annotation; visual features; color model; object detection

1. INTRODUCTION

Object detection is an important step towards automatic image understanding. Almost all current object detection frameworks require a training and validation set with object annotations. The most common form of object annotations are labeled rectangular regions. However, in many cases manual object annotations are not (sufficiently) available for training, especially in real world datasets. Labeling them manually is very tedious. In some cases it is even inacceptable due to the nature of the positive class, e.g. for adult images which are needed for training a filtering algorithm. Contrary to these difficulties, it is relatively easy today to collect positive images for many object classes via the Internet. Our goal is to determine object annotations automatically from such positive image sets.

Therefore, we present an approach to automatically determine object annotations by finding regularities among images in a set of positive images (images showing instances of the desired object class), which are not present in a set of negative images (images without instances of the desired object class). We only assume a single global binary label per image. Our motivation is to provide a preprocessing step for common object detection frameworks, which require object annotations in the form of rectangular bounding boxes.

We use a two-stage algorithm for initializing and then narrowing down bounding boxes within a set of positive images. The initializing step of our algorithm finds regions of interest (ROI) based on a discriminative color model and discriminative visual words computed from gradient features. The second step builds bag-of-visual-words histograms from the initial ROIs and trains a SVM model which is then re-applied to the training set in order to improve the bounding boxes.

Figure 1 shows an example result in order to illustrate our goal and the two stages of our algorithm. In the left image, the initialization of our algorithm by a coarse color model and a gradient feature-based model is shown. The right image shows the improved bounding box found by our algorithm.

Since we do not use manual annotations, we require all instances of the wanted object to have visual features in common, namely color and gradient features. Also, the negative dataset must be a representative background dataset, e.g. randomly downloaded images from the Internet.

We conduct experiments on six classes of the FlickrLogos-32 [10] dataset for which these assumptions hold. Also, we evaluate our approach on the Oxford 17 Flowers dataset [8].

2. RELATED WORK

As mentioned in the introduction, our algorithm consists of two main stages. The first stage is an initialization step which includes building a color model from binary image labels. The color model itself is a color histogram which corresponds to the model suggested by Jones and Rehg [6]. However, since Jones and Rehg construct their histogram from pixel-wise annotations which we do not have, we follow the approach by Ries and Lienhart [9] which only requires weakly labeled data (i.e. binary image labels).

The second stage of our algorithm utilizes bag-of-visual-words histograms as for instance proposed in [3]. These histograms are built from clustered histograms of oriented gradients (HOG) which are recently among the most popular local features. HOG features were first introduced in [1].

Since our algorithm requires rapid exhaustive maximum search within our training images based on visual-word histograms, we propose using the efficient sub window search algorithm (ESS) by Lampert [7]. The ESS algorithm is based on linear support vector machines (SVM) which we implement using SVMLight by Joachims [5].

Our algorithm was inspired by the recent state-of-the art object detection algorithm by Felzenszwalb et al. [4]. Felzenszwalb et al. also try to improve the regions of interest within the positive training images iteratively. However, they aim for correcting inaccuracies among the readily provided manual annotations while we try to find regions of interest from scratch.

3. PROBLEM AND APPROACH

Given a representative set P of images with instances and a set N of images without instances of a given object class, we aim to find a bounding box for each image of the positive dataset P which describes the position of the instance of the object class all images in P have in common. We do not assume any further annotations. However, a few assumptions about the object instances must hold: (1) The objects must have common visual features across the positive images, which cannot be found in the negative set. (2) In the positive images the background must be visually more diverse than the object instances.

For our approach these features are color and image gradients. Thus the objects must have a distinct color scheme and some visual structures in common, which we can catch with gradient-based features. Our negative images may be random images from an image repository as a few noisy images will not affect our approach.

Our goal is to estimate a rectangular bounding box $\hat{r}_i$ for each image i in the positive image set. A bounding box $r_i = (x_i, y_i, w_i, h_i)$ is a rectangle with upper left corner $(x_i, y_i)^T$, width w_i, and height h_i, specified in image coordinates. For simplicity we only search for the single best rectangle $\hat{r}_i$ in each image i, even if the image shows multiple instances of the wanted object. However, all of the methods we use can be extended to multiple instances per image.

Let r_i be the optimal (i.e. the ground truth) rectangle in image i from set P. Then, our goal is to find an rectangle $\hat{r}_i$ which maximizes the overlap $o(\hat{r}_i, r_i)$ between $\hat{r}_i$ and r_i for all images:

$$o(\hat{r}_i, r_i) = \frac{\hat{r}_i \cap r_i}{\hat{r}_i \cup r_i} \tag{1}$$

This overlap definition is a common quality measure for bounding boxes and is for instance used in the Pascal VOC challenge [2]. If an image shows multiple object instances, r_i is considered to be the one instance with which our estimation produces the best overlap. Remember that the ground truth r_i is only used for evaluation.

Our approach to this problem consists of two major stages. We first find initial bounding boxes based on color and gradient features. We thus estimate a color model for the wanted object class from image set P and then use it to create a coarse region of interest (ROI) for each image in P. Based on these ROIs, we determine local gradient features which are likely to describe the wanted object the same way we determined positive colors. The bounding boxes are then narrowed down by combining the responses of both models.

Afterwards, we train a linear SVM model on Bag-of-Visual-Words histograms for these bounding boxes. This model is used to further improve the ROIs within our images. For this purpose, we iteratively re-apply the trained SVM to the training data in order to get better training examples from improved bounding boxes for the following iteration. The following sections describe each individual step in detail.

4. INITIALIZATION

The first step of our algorithm is finding initial bounding boxes for each positive training image. We first apply a color model for a coarse first estimation and then use HOG features to improve these estimations.

4.1 Color models from global image labels

We first initialize our regions of interest based on a color model. Thus, we first need to build a model which discriminates positive colors (object colors) from negative colors (background colors). In other words, we need a binary decision function $h_c(\mathbf{p_j}) \in [0, 1]$ which determines for each pixel $\mathbf{p}_j$, if its color c_j belongs to the wanted object.

Usually, such color models are either parametric functions on some color space or histograms of color occurrence frequencies [6] which are then thresholded. We chose the latter variant (with 32^3 bins in YCbCr color space), since color histograms provide a detailed partitioning of the color space at the expense of requiring more storage space. In [9], Ries and Lienhart describe a method for creating a color histogram-based model from global image labels only. Thus, we do not have to violate our assumption about not having manual annotations of regions of interest.

The main idea of [9] is to statistically determine which colors appear regularly in positive images and less regularly in negative images. Let $f_N(c)$ and $f_P(c)$ be the relative numbers of negative images and positive images in which color c occurs. The underlying assumption is that the relative occurrence frequency $f_N(c)$ of a color c in a set of random negative images is representative for any given set of images which do not feature any particular common object. Thus, the occurrence of the respective color in any random background image is assumed to be normally distributed around the relative occurrence $f_N(c)$ observed in the negative images. If a color not only appears as a background color in a set of positive images, it will occur significantly more often as expected and therefore can be considered a positive color.

Thus, if $f_P(c) < f_N(c)$, color c is considered a negative color. Otherwise, the probability $P(f_P(c)|\neg object)$ of c being a negative color if it is observed in $f_P(c)$ of the positive images is compared against a threshold θ_c. The probability $P(f_P(c)|\neg object)$ is a normal distribution over $f_P(c)$ with $\mu = f_N(c)$ and $\sigma^2 = f_N(c)(1 - f_N(c))$. Thus, the larger the difference between $f_N(c)$ and $f_P(c)$ the smaller $P(f_P(c)|\neg object)$.

As suggested in [9] , we use the 0.97 quantile for θ_c, i.e. we adaptively select the top colors for each class. This threshold is relatively strict, however, a flood fill algorithm (seeded at pixels where $h(c) = 1$) is used on the positive images in order to extend the color model to chromatically and spatially related colors. Finally, we can use our model to determine for a given pixel $\mathbf{p}_j = (x_j, y_j)^T$ with color c_j within an image if it is a positive pixel:

$$h_c(\mathbf{p}_j) = \begin{cases} 1 & \text{if } P(f_P(c_j)|\neg object) < \theta_c \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Note that for simplicity, we re-wrote the respective decision function h from [9] by aggregating all constants into θ_c and by making it a function on a pixel $\mathbf{p}_j$.

Now we have a set $X_i = \{\mathbf{p}_j | h_c(\mathbf{p}_j) = 1\}$ of color-based positive pixel locations in a given image i. Based on this set we compute an initial bounding box as described in the following section.

4.2 Finding Bounding Boxes based on positive pixels

After creating an initial model which assigns a binary value to each pixel within an image, we can use the positive pixels to form a bounding box. We assume that the majority of positive pixels is located on the wanted object and thus close to its center while only a few are at a large distance of the object. Therefore, we expect the distribution which generated the positive pixels to be a normal distribution centered at the wanted object's centroid. We thus fit a normal distribution into the two-dimensional distribution of positive pixels as follows.

We first compute the center pixel $\mu_i = (\mu_{i,x}, \mu_{i,y})^T$ of the object as the mean value of all positive pixels X_i and the corresponding standard deviations $\sigma = (\sigma_{i,x}, \sigma_{i,y})^T$. Our bounding box coordinates $\hat{r}_i = (x_i, y_i, w_i, h_i)$ are then

$$\begin{aligned} x_i &= \mu_{i,x} - \sigma_{i,x}(1+\beta) \\ y_i &= \mu_{i,y} - \sigma_{i,y}(1+\beta) \\ w_i &= 2\sigma_{i,x}(1+\beta) \\ h_i &= 2\sigma_{i,y}(1+\beta) \end{aligned} \quad (3)$$

The factor β is used to enlarge the bounding box. According to an exhaustive parameter sweep, $\beta = 0.6$ is a good choice which we use for all experiments.

If the color model fails and does not find any positive pixels, we use the full image for the following step, since we know the object is present at some location in the image.

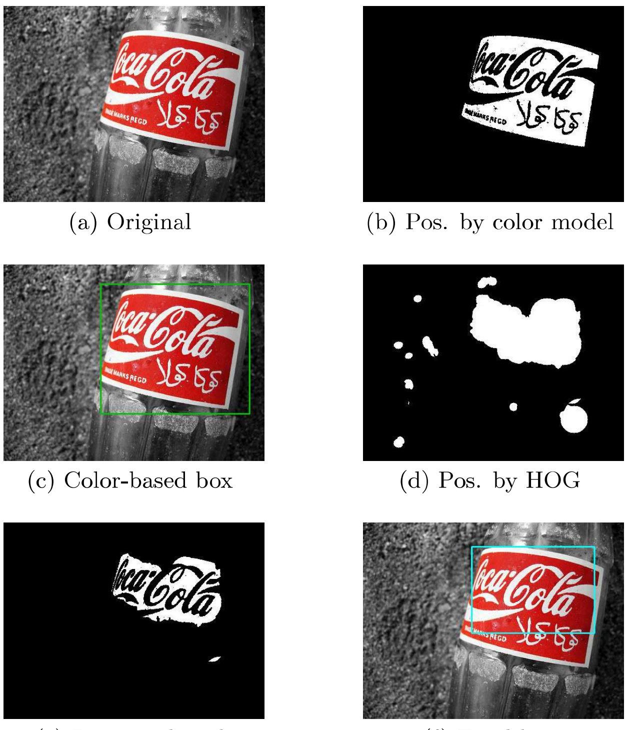

(a) Original (b) Pos. by color model (c) Color-based box (d) Pos. by HOG (e) Pos. combined (f) Final box

Figure 2: Example for the initial bounding boxes. (a) shows the original image, (b) the positive pixels found by the color model as white pixels, (c) the resulting bounding box based on color model, (d) the positive pixels determined by HOG features, (e) the combined model (i.e. "AND" between (b) and (d)), (f) final bounding box based on combined model.

Also note that we could use the same method for creating multiple bounding boxes, by fitting a Gaussian mixture with multiple peaks into the two-dimensional distribution of positive pixels using an EM algorithm. In this paper however, we only examine the single-object problem.

While the color model described in the previous section overall correctly identifies most positive pixels and a large portion of negative pixels, it does not yield very accurate regions of interest for multiple reasons. First, positive colors naturally also appear in the background. Second, for some classes (e.g. brand logos) the actual wanted object may be surrounded by a background area of the same color. Also, there are obviously always a few noisy colors, i.e. false positive and false negative colors. Thus, if we use the color model to find a ROI as described in the following section, we usually "overdetect" the object.

In figure 2, an example for the abovementioned problem is shown. The wanted object (the brand logo) is detected together with a larger background area due to false positive detections (see figures 2b and 2c) which are difficult to avoid by a color model. We therefore utilize a second feature which is based on gradients in order to improve our initialization.

4.3 HOG-based models

Since the color model often produces false positive regions, we add another feature to our initialization process. Histograms of oriented gradients (HOG) [1] model edge-based information and are thus complementary to our color model. We use an implementation of HOG which is analogous to the implementation described in [4]. The HOG descriptor is usually computed on a dense grid of cells. We set the cell size to 8 pixels and concatenate groups of four neighboring cells into a single feature vector in order to increase the expressiveness of a single feature as suggested in [1]. Since we do not know the relative size of the wanted object within a positive image, we extract HOG on multiple scales on a scale space pyramid with scaling factor $(2^{-0.5})^{-0.5}$.

We now use the HOG features in the same way as the color model, thus we need to limit the infinite set of potential HOG features to a finite number of identifiable features. This can be done by a Bag-of-Visual-Words (BOW) model.

A BOW model requires a Visual Dictionary which is a finite set of prototypical visual features, the so-called visual words. The visual words are created by extracting a large number of visual features and clustering these features into k groups. Each cluster is then represented by its cluster id, thus the number of different local features is reduced to k. For our experiments, we cluster the HOG features into $k = 1,000$ visual words.

Since we now have a limited number of different features, we can apply the same strategy we use for the initial color-based bounding boxes. We simply count the relative numbers $f_P(w)$ and $f_N(w)$ of positive and negative images with a given visual word w. However, since we are confident, that the color model is very likely to return a region which includes the actual object, we only count images for $f_P(w)$ where w is found inside the color-based bounding box.

We then compute probability $P(f_P(w)|\neg object)$ analogous to $P(f_P(c)|\neg object)$ in section 4.1 and use an equation which is analogous to equation 2 in order to determine if a HOG grid point $\mathbf{p}_l$ is a positive pixel:

$$h'_w(\mathbf{p}_l) = \begin{cases} 1 & \text{if } P(f_P(w_l)|\neg object) < \theta_w \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where w_l is the visual word observed at grid point $\mathbf{p}_l$. Since from h'_w we do not obtain a decision at every possible pixel location j, we define

$$h_w(\mathbf{p}_j) = \begin{cases} 1 & \text{if } \exists l : h_w(\mathbf{p}_l) = 1 \wedge \parallel \mathbf{p}_l - \mathbf{p}_j \parallel^2 < w_{hog} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where w_{hog} is the width of one HOG cell and $\mathbf{p}_l$ is a HOG grid point for which $h_w(\mathbf{p}_l)$ can be determined as described above. This provides us with a binary map of positive and negative pixels analogous to the color model.

In contrast to the color model, however, we must choose a less strict threshold θ_w since we cannot use an additional flood fill algorithm on the HOG features. Besides, the number of different HOG features is far smaller than the number of different colors. Thus, we use an adaptive threshold which depends on the lowest value p_{best} of $P(f_P(w)|\neg object)$ found for any word w. Empirically, we define $\theta_w = 20 \cdot p_{best}$, i.e. we use all visual words within a certain interval defined by the best word with regards to their probability. Note that this strategy ensures that if the visual words are not very characteristic for a given class (i.e. if p_{best} is large) we will select a large number of visual words. Thus, a "bad" set of visual words results in large areas of positive pixels which are neutral for the combined model explained below and do not produce false negative pixels.

4.4 Combined Initialization

As mentioned above, our assumption is that the wanted objects are characterized by both, distinct colors and distinct gradient features. Therefore, we expect to find positive colors as well as positive visual words on the wanted object. We thus now determine locations, where both are present by applying a decision function which combines color and gradient evidence at each pixel $\mathbf{p}_j$ by

$$h(\mathbf{p}_j) = \begin{cases} 1 & \text{if } h_c(\mathbf{p}_j) \wedge h_w(\mathbf{p}_j) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We now use updated pixel sets $X'_i = \{\mathbf{p}_j | h(\mathbf{p}_j) = 1\}$ to create new bounding boxes as described in section 4.2.

In figure 2, all stages of the initialization process are shown. Figures 2b and 2c show the positive pixels determined by color model h_c and the resulting bounding box. Figures 2d and 2e show the positive pixels determined by the hog feature model h_w and the combined model h. Finally, figure 2f shows the bounding box based on the positive pixels of the combined model. Another example for the three components of the initial model is shown in figure 1 (left).

The combined model stil tends to detect false positive regions. We thus try to further improve our bounding boxes with an iterative algorithm explained in the next section.

5. ITERATIVE ALGORITHM

In the previous sections we have explained how we create an initial bounding box for each of our positive images based on occurrence statistics of colors and HOG features. We now want to show that we can use these initial bounding boxes for a discriminative, iterative algorithm in order to further improve the bounding boxes.

5.1 Representation of examples by BOW histograms

At first glance, the nature of the classes we use for our experiments (e.g. brand logos) indicates the usage of rigid templates which are applied (enhanced by deformable geometric layouts) by state-of-the-art object detection algorithms, for instance [4, 11]. Such templates basically consist of given aspect ratios of horizontal and vertical cells.

However, we do not use manual annotations for regions of interest and our initial bounding boxes which are created by the initial models are often highly noisy. Thus, no meaningful single aspect ratio can be selected because we cannot expect to find similar HOG cells at similar relative grid locations. Since the initial bounding boxes often deviate significantly from the actual object position, we also cannot expect to learn a meaningful part representation.

These problems can be sidestepped to some extent (at the expense of a much less specific model) by using BOW histograms. A BOW histogram is an occurrence histogram over the k clusters of our visual dictionary within a ROI. We L1-normalize all BOW histograms in order to obtain relative occurrence frequencies.

Therefore, bounding box $\hat{r}_i$ of image i is represented by a k-dimensional relative word occurrence histogram vector $\mathbf{x}_i$. Thus, it is guaranteed that each bounding box is represented by a vector of the same dimensionality k which is independent of the actual number of cells within the bounding box.

However, we still limit the set of aspect ratios we use to model examples to a reasonable set. At the same time, we only want to have training examples with approximately the same number of HOG cells for comparability. We therefore use a template set T consisting of templates for which we define a constant minimum width and height $c \in \mathbb{Z}$ and a maximum width and height $1.5c$:

$$T = \{(w,h) | w,h \in [c, 1.5c] \wedge w \cdot h \approx c \cdot 1.5c\} \qquad (7)$$

As a default value, we choose $c = 8$. If we assume that additional information about the absolute size of the images or the usual relative sizes of the wanted objects is available, we can adjust c accordingly. For instance, for a dataset which only consists of images with relatively large objects (e.g. Oxford 17 Flowers) it is reasonable to increase c.

Note that vector $\mathbf{x}_i$ can only be created for a template rectangle which lives on the multi-scale HOG grid. The initial rectangle $\hat{r}_{i,h}$ created by our initial model h on the original image scale, however, lives in image coordinates which usually do not match the HOG grid. Thus, for representing an arbitrary bounding box $\hat{r}_{i,h}$ by a bounding box $\hat{r}_i$ on the multi-scale HOG grid, we find the template rectangle $\hat{r}_i$ in scale space which best matches $\hat{r}_{i,h}$ with regards to overlap. Histogram vector $\mathbf{x}_i$ is then computed from $\hat{r}_i$.

For our negative examples, we randomly sample valid bounding boxes (i.e. rectangles which fit one of our templates on a random scale) from a set of m negative images.

Due to the inaccurate initial bounding boxes, all positive examples include a number of noisy (negative) visual words. We thus use a linear SVM classifier which is to some extent robust to noisy input data. We expect our SVM classifier to learn which visual words are found in most positive examples, while the remaining (non-object) words appear more at random since they originate from background areas.

Another advantage of BOW models and linear SVMs is that they allow applying an efficient algorithm for multi-scale subwindow search.We propose using the Efficient Subwindow Search (ESS) algorithm introduced by Lampert et al. [7] which is explained in the following section.

5.2 Training and Detection

In this section we explain the classification method we use and how we efficiently search for the best rectangles within the positive images during each iteration. In the first iteration of our algorithm, we perform SVM training on the bounding boxes found by the initial model. For the remaining iterations, we train our current new model on the best detections found by the previous model.

The iterative algorithm terminates after a given number of iterations or if the bounding boxes do not change anymore between subsequent iterations. For our experiments, we only use two iterations since our linear classifier does not improve beyond a few iterations. The reason for applying the iterative algorithm, however, is mainly to show that our initial models deliver bounding boxes which can be used to train a discriminative classifier in an iterative fashion to further improve the bounding boxes.

5.2.1 *Training and Efficient Classification*

We have a set of n positive BOW histograms $\mathbf{x}_i, i \in \{0, ..., n\}$ and m negative histograms $\mathbf{x}_j, j \in \{0, ..., m\}$. We now need to devise a decision function $f(\mathbf{x}_l)$ which returns a score value for an unknown example $\mathbf{x}_l$ (i.e. the BOW representation of an arbitrary rectangle r_l of image l) indicating either positive or negative classification. Since we need to perform an exhaustive search for the best rectangle in an image, we propose using an efficient search algorithm analogous to the approach by Lampert et al. [7] which requires a linear decision function. Thus, for $f(\mathbf{x}_l)$ we use a linear SVM model which is defined as

$$f(\mathbf{x}_l) = \beta + \sum_k \alpha_k \langle \mathbf{x_l}, \mathbf{x}_k \rangle \qquad (8)$$

where $\mathbf{x}_k$ denotes the k-th training example (including positive and negative examples), $\langle \cdot, \cdot \rangle$ is the scalar product, and α_k and β are constant weights and bias which have to be learned. We use the SVM implementation by Joachims [5] for learning the optimal α-weights. Note that for some training examples $\alpha_k = 0$ since they are not selected as support vectors during training. The bias β may be neglected since it does not influence the maximum search explained below.

Each visual word v then obtains an individual weight w_v:

$$w_v = \sum_k y_k \alpha_k x_{k,v} \qquad (9)$$

where $x_{k,v}$ is the v-th entry of BOW histogram $\mathbf{x}_k$ (thus the frequency of visual word v in the $k - th$ training example), and $y_k \in \{1, -1\}$ is the respective image's label.

Following Lampert et al., we now re-write the linear SVM function 8 as a function on the respective rectangle r_l, from which $\mathbf{x}_l$ was extracted, as follows

$$f(r_l) = \beta + \sum_{\mathbf{p} \in P_{r_l}} w_{v,\mathbf{p}} \qquad (10)$$

Where $\mathbf{p}$ denotes a two-dimensional point on the HOG grid of image l and P_{r_l} is the set of all points enclosed by rectangle r_l. Note that the function $f(r_l)$ is a function which computes the sum over values within a given rectangle and can thus be computed very rapidly for arbitrary rectangles by using integral images.

5.2.2 *Branch And Bound*

Using integral images on visual word weights as described above, we can evaluate the classification score of an arbitrary rectangle with only four look-ups. Still, we have to evaluate many possible rectangles, since we perform multi-scale search with multiple templates. In order to further speed up the search, we suggest utilizing a branch and bound search strategy, also proposed by Lampert et al. [7]. The main idea is computing a maximum quality (i.e. an upper bound) for a set of potential rectangles in order to be able to dismiss a large number of rectangles at a time instead of evaluating each rectangle individually. Since we use a linear SVM, we can compute the upper bound based on per-point contributions and thus by using integral images as described above.

We first require a bounding function $\hat{f}(R)$ which is defined on a set of rectangles R and has to fulfill two conditions. First, $\hat{f}(R)$ must be an upper bound of $f(r)$. Function $\hat{f}(R)$ is an upper bound if its value is at least as large as the SVM score of the best individual rectangle within R. Second, $\hat{f}(R)$ must converge to $\hat{f}(R) = f(r)$ if R only contains r as a single element. A valid bounding function, for which both conditions hold is

$$\hat{f}(R) = f^{+}(\cup r) + f^{-}(\cap r) \qquad (11)$$

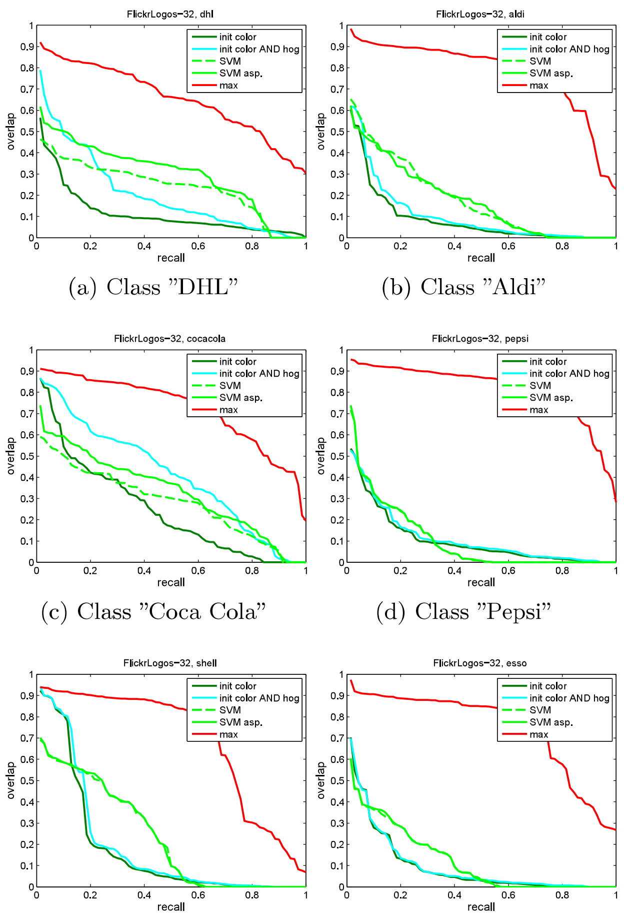
where $f^+(\cup r)$ are the positive summands of f on the union $\cup r$ of all rectangles in R and $f^-(\cap r)$ are the negative summands of the intersection $\cap r$ of all rectangles in R. It is relatively straightforward to show that $\hat{f}(R)$ is at least as large as the SVM score of the best $r \in R$ and thus that the function is a valid bound. For further details refer to [7].

Now we can efficiently search by branch and bound. Thus, we first split the set of all possible rectangles into two disjoint subsets and compute the bound of both sets. According to their bounding function values, both sets are added to a priority queue (where the set with the highest bound value is on top). Then, we iteratively repeat this process for the current top set of rectangles of the queue. When the top set of rectangles in the priority queue only consists of one single rectangle, we have found the globally optimal rectangle.

5.2.3 *Using a predefined aspect ratio*

Due to our scaling pyramid, we have much more small candidate rectangles than large ones. Hence our approach tends to prefer small boxes over large ones, since the training algorithm is more likely to select histograms from larger scales as positive examples. Also, squared boxes which only cover a principal part of the object are often preferred, since our algorithm does not explicitly prohibit finding a common partial object instead of the full wanted object.

If we however assume that we roughly know the actual aspect ratio of the wanted object, we can use this information in order to find better fitting bounding boxes based on the ones we found. Thus, we apply an heuristic which only requires defining the object as being ”horizontal”, ”vertical”, or ”squared” for each class. Based on this information, we first enlarge the bounding box found by our SVM by 10% at each edge and then transform the box into a horizontal or vertical bounding box while preserving the area of the box. For horizontal and vertical boxes we simply use an aspect ratio of $\frac{2}{1}$ or $\frac{1}{2}$, respectively. For the ”squared” case we assign a squared bounding box with roughly the same area as the original rectangle.



(a) Class ”DHL” (b) Class ”Aldi”

(c) Class ”Coca Cola” (d) Class ”Pepsi”

(e) Class ”Shell” (f) Class ”Esso”

Figure 3: Overlap-recall curves for all six logo classes.

6. EVALUATION

As mentioned in section 3, our goal is to find an optimal bounding box with regards to overlap as defined in the problem statement in equation 1. Thus, for evaluation we use rectangular ground truth annotations r_i for each positive image i. Given this annotation, we can compute an overlap value for our estimated bounding box $\hat{r}_i$ for each image. This enables us to plot an overlap-recall (OR) curve which shows the relative number of images (the recall) in which the overlap of our estimation surpasses a given value. Since we only aim to find a single best bounding box for each image, only the best overlapping ground truth rectangle is counted for our evaluation.

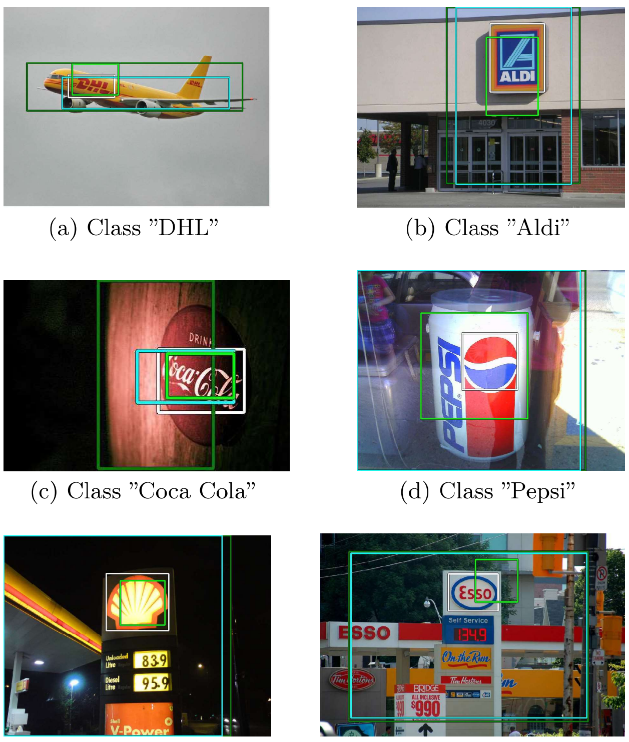
Each plot shows the result after applying our color model (labeled as ”init color”). The results for the combined initial model of color and HOG (equation 6) are shown as ”init color AND hog”. If we do not find a bounding box within an image, we count overlap 0 for the respective model and image. Note that in a detection scenario, one could simply use the full image as a bounding box in this case in order not to miss the object (given the fact that we already know the image is positive). However, the overlap is then completely independent from the quality of the model and depends from the size of the object only.

For clarity, we only show the result after the final iteration of our iterative algorithm in our OR curves, labeled as ”SVM”. Each plot also provides the final result we obtain if we apply our aspect ratio heuristic (see 5.2.3) after each iteration (labeled ”SVM asp”).

For each class, we also plot the best possible result labeled as ”max”. This curve shows the best overlap recall curve we could reach if we always found the best overlapping rectangle for each image in the search space. Note that a perfect overlap score of 1.0 is very unlikely to be obtainable since the annotations do not live on the HOG cell grid and also the number of our templates (i.e. the different aspect ratios we use for searching) is limited. Also note that the initial models are not affected by these limitations.

6.1 FlickrLogos-32

We first test our approach on the same six classes from FlickrLogos-32 [10] used in [9]: ”DHL”, ”Aldi”, ”Shell”, ”Esso”, ”Coca Cola”, and ”Pepsi”. These classes are selected in [9] since they all have a distinct color scheme. Each class consists of 70 positive images and FlickrLogos-32 also provides a negative set with 6, 000 images which we all use for creating the initial model. For SVM training, we sample 5 random bounding boxes (based on our set of templates) as negative


(a) Class "DHL"

(b) Class "Aldi"

(c) Class "Coca Cola"

(d) Class "Pepsi"

(e) Class "Shell"

(f) Class "Esso"

Figure 4: One example result for each logo class. The dark green rectangle indicates the bounding box found by the color model, cyan shows the initial combined model, and light green is the final result of our algorithm. The white rectangle is the respective ground truth annotation which is used to compute the overlap for our OR curves.

examples from each negative image. Since we hence have a very unbalanced dataset, we weight the training examples proportionally by factor m/n for the SVM training where m is the number of negative examples and n is the number of positive examples. In figure 3, we show our resulting OR curves. A few qualitative examples are shown in figure 4.

The combined initial model outperforms the individual color model for classes "DHL", "Aldi", and "Coca Cola". For the remaining classes, the combined model is identical to the color model, thus the initial HOG model is relatively poor and does not contribute to reducing background areas.

Also, our iterative algorithm further improves the rectangles significantly in comparison to the initial bounding box for all logo classes except "Pepsi" and "Coca Cola". Interestingly, for "Coca Cola" our combined initial model performs even better than the iterative algorithm, since the visual words describing the Coca Cola logo are very specific and the initial models are not bound to the HOG grid. The aspect ratio heuristic slightly improves the results for all classes, which are not defined as "squared" ("DHL", "Coca Cola", and "Aldi" are "horizontal" or "vertical").

For the class "Pepsi" our approach fails to improve the overlap beyond the initial models. The Pepsi logo is visually not as characteristic as the other logos with regards to HOG features. It consists of two plain uniformly-colored areas with relatively soft, curved edges. Therefore the edges (and thus gradients) do not provide much more information than the colors. Also the Pepsi logo comes in two different variants and is often small and rotated in the training data.

Obviously, our results are not even close to perfect detection. However, note that the overlap score we use quickly diminishes with slight deviations from the ground truth rectangle. For instance, the example shown in figure 4b has an overlap of only 0.49 although it is subjectively a reasonable detection. Only examples 4a and 4e slightly surpass an overlap of 0.5, and only 4c (scarcely) reaches 0.8. For this reason, the widely acknowledged Pascal VOC challenge [2] (from which we adopted the overlap criterion) defines an overlap of 0.5 or higher as a correct detection.

Also note that a small overlap indicates that at least the general position of the object may have been found.This observation is confirmed by the fact that estimating the correct aspect ratio improves the results for some classes. Similarly, small overlap values are also produced by strong overdetection, however it is unlikely for our SVM results due to the limited sizes of our templates.

Given the fact that we do not utilize any manual annotations or additional information except for binary image labels and one general aspect ratio, we think that our results are promising and that the approach is worth further research.

6.2 Oxford Flowers

For our second experiment (also analogous to [9]), we use the Oxford 17 Flowers [8] dataset. This dataset provides 17 different flower classes, each of which consists of 80 positive images. Since no negative set is provided, we adopt the strategy of [9] and simply use all 16 remaining classes as negative images for a given positive class.

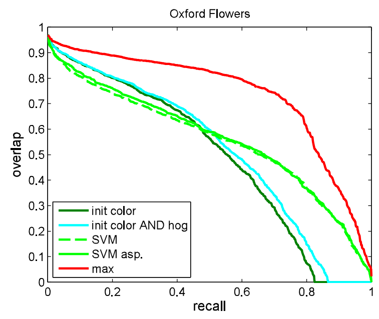


Figure 5: Overlap-recall curve over all (annotated) images from Oxford Flowers dataset.

The Flowers dataset comes with pixel annotations. Since for our evaluation we need ground truth bounding boxes, we use a blob detector on the pixel annotations and fit a rectangle around each blob we find. Note that if two instances of an object overlap in an image, we will hence only obtain one single bounding box surrounding both instances. Also note that the number of annotations per class strongly varies (one class even has none).

Our experiment is conducted analogously to the experiment on FlickrLogos-32. Since the flower images are smaller than the logo images, however, we consider less scales for our multi scale search. Also, the negative set is much smaller, so we double the number of random instances per image.

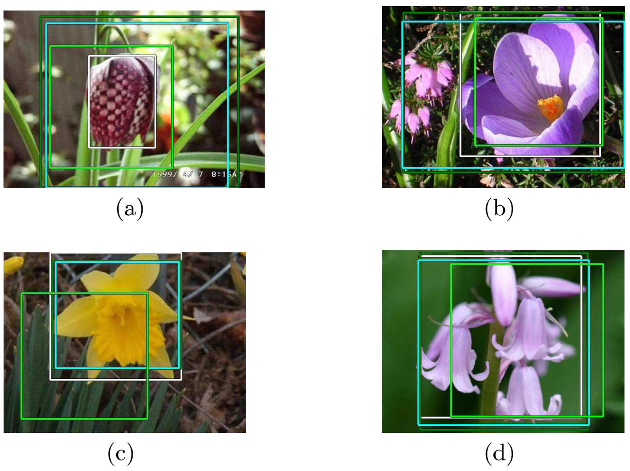


Figure 6: A few example results for the Oxford Flowers dataset. Rectangle colors are analogous to figure 4.

Since we have a large number of classes, we do not plot each class separately but combine them all into one single plot which is shown in figure 5. The results show that we increase the overlap with the ground truth in comparison to the initial model, but there is still room for improvement. Again, the HOG model does not contribute much to the combined model since the positive HOG features on the flower images are naturally less different from the background than for brand logos. Also, for most flower classes color is a stronger feature than gradients.

The results also show that our iterative algorithm is slightly worse than the combined initial model for the top overlaps (on the left of the plot) which is due to the fact that the initial model is not bound to the rigid HOG grid. However, applying the algorithm still improves the overall performance clearly since many objects are detected which are missed by the initial model as the right half of the plot indicates.

Again, we also show a few qualitative results in figure 6. In 6a and 6b our algorithm improves the detection in comparison to the initial models. Figures 6c and 6d show examples where our algorithm fails to improve the bounding box beyond the color model. For 6c, one can see that there are structures in the background which closely resemble the actual respective object. Thus, the HOG model cannot distinguish the background from the foreground while the color model is not affected by this type of noise.

7. CONCLUSION AND FUTURE WORK

In this paper we have suggested an approach to automatically creating rectangular annotations for a given set of images which feature a common object. We only use binary image labels, i.e. we are only given positive and negative sets of images. Our approach then aims to find regions of interest for all positive images.

We propose an algorithm which starts with coarse initial bounding box estimations and iteratively improves these boxes based on a bag-of-visual-words model. During each iteration, the algorithm performs an exhaustive search for the current best bounding box which can be done efficiently by using a linear SVM model and branch and bound. Based on the bounding boxes found, the model is then re-trained.

Our results are promising given the fact that we do not utilize any previous knowledge except binary image labels (and a coarse estimation of the aspect ratio). Yet there is still much room for improvement which is also shown by our evaluation. Thus, there are many options for future work.

For instance, we think that our initial models can be refined and extended in order to produce better initial estimations. Also, we can enrich our BOW model by geometric information by using a spatial layout which can also be seamlessly integrated into our search algorithm as shown by Lampert et al. [7]. Besides, our detection algorithm is still very basic. Adopting more ideas from state-of-the-art object detection algorithms, such as latent models, may also improve our approach if adjusted to accept noisy initial bounding boxes as an input instead of manual annotations.

Also, our test classes are relatively easy, since the wanted objects are homogenous and mostly rigid, so another important aspect of future work will be dealing with more difficult classes and detecting multiple objects per image.

8. REFERENCES

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE CVPR05*, volume 1, pages 886–893 vol. 1, Jun. 2005.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge. *IJCV*, 88(2):303–338, Jun. 2010.

[3] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE CVPR05.*, volume 2, pages 524–531 vol. 2, Jun. 2005.

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

[5] T. Joachims. Advances in kernel methods. chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[6] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *IJCV*, 46(1):81–96, Jan. 2002.

[7] C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE PAMI*, 31(12):2129 –2142, Dec. 2009.

[8] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *IEEE CVPR06*, volume 2, pages 1447–1454, 2006.

[9] C. X. Ries and R. Lienhart. Deriving a discriminative color model for a given object class from weakly labeled training data. In *ACM ICMR12*, pages 44:1–44:8, New York, NY, USA, 2012. ACM.

[10] S. Romberg, L. G. Pueyo, R. Lienhart, and R. van Zwol. Scalable logo recognition in real-world images. In *ACM ICMR11*, pages 25:1–25:8, New York, NY, USA, 2011. ACM.

[11] L. Zhu, Y. Chen, A. L. Yuille, and W. T. Freeman. Latent hierarchical structural learning for object detection. In *IEEE CVPR 2010*, pages 1062–1069, 2010.