Fast and Robust Object Segmentation with the Integral Linear Classifier
David Aldavert, Arnau Ramisa, Ricardo Toledo, Ramon Lopez de Mantaras

To cite this version:
David Aldavert, Arnau Ramisa, Ricardo Toledo, Ramon Lopez de Mantaras. Fast and Robust Object Segmentation with the Integral Linear Classifier. CVPR 2010 - 23rd IEEE Conference on Computer Vision & Pattern Recognition, Jun 2010, San Francisco, United States. pp.1046-1053, 10.1109/CVPR.2010.5540098. inria-00548642

HAL Id: inria-00548642
https://inria.hal.science/inria-00548642
Submitted on 20 Dec 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Fast and Robust Object Segmentation with the Integral Linear Classifier

David Aldavert
Computer Vision Center
Dept. Computer Science
Univ. Autònoma de Barcelona

Arnau Ramisa
INRIA-Grenoble
Artificial Intelligence Research Institute (IIIA-CSIC)

Ramon Lopez de Mantaras
Artificial Intelligence Research Institute (IIIA-CSIC)
Campus UAB

Abstract

We propose an efficient method, built on the popular Bag of Features approach, that obtains robust multiclass pixel-level object segmentation of an image in less than 500ms, with results comparable or better than most state of the art methods. We introduce the Integral Linear Classifier (ILC), that can readily obtain the classification score for any image sub-window with only 6 additions and 1 product by fusing the accumulation and classification steps in a single operation. In order to design a method as efficient as possible, our building blocks are carefully selected from the quickest in the state of the art. More precisely, we evaluate the performance of three popular local descriptors, that can be very efficiently computed using integral images, and two fast quantization methods: the Hierarchical K-Means, and the Extremely Randomized Forest. Finally, we explore the utility of adding spatial bins to the Bag of Features histograms and that of cascade classifiers to improve the obtained segmentation. Our method is compared to the state of the art in the difficult Graz-02 and PASCAL 2007 Segmentation Challenge datasets.

1. Introduction

Currently, some of the most successful approaches to object categorization are based in bags of features (BoF) [8]. In spite of its simplicity, this type of methods have demonstrated very good performance in many state of the art object categorization datasets [4, 19], and in other tasks such as place recognition [3, 21] or texture classification [25, 11]. However, since the distribution of the local descriptors is lost in the accumulation step, it is impossible to localize the object of interest in the image and the output of the classifier is restricted to a binary present/not present label.

The most straightforward solution to overcome this limitation is to use sliding windows [24, 23], which is a generalist strategy that has no pre-assumptions on the underlying data and does not depend on the performance of a given segmentation algorithm. However, in order to perform an exhaustive sampling of the window space, millions or even billions of possible windows have to be considered, which turns the problem intractable from a computational point of view. A solution to the window search problem, with only some restrictions on the histogram normalization, was found by Lampert et al. [9], who proposed to use a branch and bound schema over the window space to direct the search and dramatically reduce the computational effort to find the optimal sub-window. Another drawback of the sliding windows approach is that only an approximate localization can be obtained, and there is no guarantee that the best classified window will contain the totality of the object. As an alternative to the sliding windows, in this work we advocate for pixel-level labeling [10, 6] to accurately segment class-consistent regions from the image. As in the case of sliding windows, this approach does not rely in the performance of a particular image segmentation algorithm, but region boundaries are naturally obtained from the object class conditional probabilities of every pixel. Although many variations of the base BoF method have been proposed, all of them consist on four basic steps. Namely, feature extraction from the image, feature quantization into visual words, accumulation of visual words in histograms and classification of the resulting histograms. Nevertheless, labeling every pixel of an image is a computationally demanding task, that requires carefully optimizing each step of the method if a fast implementation is required. For example, in [6], the authors propose an efficient method for each step by using a hierarchical k-means codebook [17],
and using integral images (both for feature extraction and accumulation into histograms) with optimized small codebooks. Nevertheless, in their approach a complete integral image is necessary for each dictionary word, and the resulting histogram still needs to be classified.

Here we propose a very efficient BoF-based approach that uses the novel Integral Linear Classifier to determine the classification of an image sub-window, merging both accumulation and classification in a single step. We also contribute an evaluation of the performance and computational cost of different alternatives for dense feature extraction and codebook generation. The performance of the method is evaluated in the task of assigning a category label to each pixel of the image in the GRAZ02 dataset and the PASCAL 2007 Segmentation Challenge (VOC2007) dataset.

2. Efficient Pixel Categorization

A BoF approach is used to efficiently calculate the object class probabilities for every pixel\(^1\). First, densely sampled region descriptors are quantized into visual words using a codebook. Then, a fast sliding window approach is used to assign the classwise probabilities to every pixel: visual words within a window are accumulated in a histogram, which is later categorized using a linear support vector machine. Next, we show different existing alternatives to efficiently execute all the previous steps, and how accumulation and categorization can be fused into a single operation when a linear classifier is used.

2.1. Efficient Dense Features

Using densely sampled image descriptors as input data has several advantages when compared to keypoint-based approaches: more information can be extracted from the underlying image and the time-consuming keypoint detection step is not needed [18]. Furthermore, when image descriptors are not rotation invariant, their computation can be speeded up using integral images, which can easily make this approach even faster than the keypoint-based alternatives despite the larger number of descriptors computed. Although rotation invariance is a good descriptor property for wide baseline feature matching, Zhang et. al. [25] found that they have a negative effect in the performance of BoF approaches. In this paper we compare three different local descriptors that can be calculated using integral images: Speeded-Up Robust Features (SURF), Integral Histograms of Oriented Gradients (IHOG) and Integral Shape Context (ISC). These local descriptors are computed on image gradient values. We use Haar wavelets to calculate the image gradient because they are less sensitive to noise [1] and can be efficiently computed regardless the scale of the feature.

2.1.1 SURF

The SURF descriptor [1] is obtained by accumulating the sum of Haar wavelet responses at different spatial bins. Let \( d_x \) be the Haar wavelet response in the horizontal direction and \( d_y \) the Haar wavelet response in the vertical direction. Then, the SURF descriptor, with \( P \times Q \) spatial bins, is computed by adding the wavelets responses \( d_x \) and \( d_y \) over each bin sub-region. To take into account information about the polarity of the intensity images, the sum of absolute values of the responses, \( \|d_x\| \) and \( \|d_y\| \) is also extracted. Hence, each spatial bin of the descriptor has a four-dimensional descriptor vector with the form \((\sum d_x, \sum \|d_x\|, \sum d_y, \sum \|d_y\|)\). Finally, the resulting \( 4 \times P \times Q \) dimensions descriptor is L1-normalized.

\[ \begin{align*}
    \Sigma_x, & \sum \|d_x\|, \sum d_y, \sum \|d_y\|
\end{align*} \]

Figure 1. SURF descriptors for different image patches.

In our implementation of the SURF descriptor, the Gaussian weighting mask with the origin at the feature center is not used, so that, it can be directly computed from integral images. Thus, only \( 4 \times (P + 1) \times (Q + 1) \) memory accesses and \( 4 \times P \times Q \) additions are necessary to compute the descriptor, regardless of the feature region size.

2.1.2 Integral Histogram of Oriented Gradients

The Integral Histograms of Oriented Gradients (IHOG) [26] is an approximation of the popular SIFT descriptor [14] that can be efficiently computed with integral images. An IHOG descriptor with \( N \) orientation bins and \( P \times Q \) position bins, i.e. a \( N \times P \times Q \) dimensional descriptor, is calculated as follows: First, \( N \) orientation images, one for each orientation bin of the descriptor, are created interpolating the gradient orientation weighted by its module for each pixel of the image. Then, creating an integral image from these images, the value of an orientation bin can be calculated with only 4 additions. Therefore, only \( N \times (P + 1) \times (Q + 1) \) memory accesses and \( N \times P \times Q \) additions are needed for every IHOG descriptor, regardless of the feature size. Fig. 2 shows an example of the orientation and integral images used to calculate the IHOG descriptor.

Unlike the SIFT descriptor, the IHOG descriptor is incompatible with the Gaussian mask and the tri-linear inter-

\(^1\)The source code of the proposed method can be found at http://www.cvc.uab.cat/~aldavert/plor/
polation to weight the contribution of the gradient module in the spatial bins of the descriptor. Another difference is that the IHOG descriptor uses L1-norm instead of the L2-norm.

### 2.1.3 Integral Shape Context

The shape-context descriptor [2] is obtained by accumulating gradient module in log-polar distributed bins. However, when spatial bins are distributed over a rectangular grid the descriptor can be calculated using integral images. Then, the Integral Shape Context (ISC) descriptor is computed by accumulating the module of the gradient into \( P \times Q \) position bins and L1-normalizing the obtained descriptor. This descriptor can be efficiently computed using an integral image created from the gradient module. In Fig. 3, an example of how the ISC descriptor is generated from the integral of the gradient module image is shown.

![Figure 3. ISC example: The integral image b) is generated from the module of the gradient image a). Then, the ISC descriptor can be calculated from the integral image b). For example, the first descriptor dimension amounts to \( D_0 = D - B - C + A \), independently of the region size.](image)

Then, the ISC descriptor of a \( P \times Q \) position bins (i.e. a \( P \times Q \) dimensional descriptor) can be calculated with only \( (P + 1) \times (Q + 1) \) memory accesses and \( P \times Q \) additions, regardless of the feature regions size.

### 2.2. Codebook Generation

The HKM [17] defines a hierarchical quantization of the feature space. Instead of \( k \) being the final number of visual words of the codebook, it determines the branch factor (i.e. the number of descendants of each node) of a tree. An HKM is generated as follows: First, the \( k \)-means algorithm is used to split the training data into \( k \) clusters. Then, this clustering process is recursively applied to the cluster from the previous level until a maximum depth is reached or until the amount of data samples in the current cluster is lower than a given threshold. This recursive method creates a codebook with a reduced computational cost both in the training and descriptor quantization phases. The computational complexity of quantizing a \( D \) dimensional descriptor using a HKM with \( V \) visual words is \( O(Dk \log_k V) \). In the original implementation of the HKM, all nodes of the tree are used as visual words to alleviate misclassification problems in the superior levels of the tree and the contribution of each node of the histogram is weighted by \( w_i = \ln \left( \frac{N}{n_i} \right) \), where \( n_i \) is the number of BoF histograms that contains the visual word \( i \), and \( N \) is the total number of BoF histograms.

However, the use of these two refinements have a modest impact in the performance of the HKM. Therefore, they are removed from our implementation.

The ERF [16] uses a combination of several random K-D trees in order to quantize the feature space. Each K-D tree of the ERF is built recursively in a top-down manner as follows: Given a set of labeled training descriptors (i.e. descriptors with a category label associated), the K-D tree splits the training descriptors from the previous level into two disjoint sets with a Boolean test in a random bin \( D_t \) and a random threshold \( \theta_t \). For each split, parameters \( (\theta_t, D_t) \) are generated randomly until a pair that attains an entropy value larger than \( S_{\text{min}} \) is found, or the maximum number of trials \( T_{\text{max}} \) has been reached. The parameter \( S_{\text{min}} \) can be adjusted to select the desired randomness of the K-D trees. For instance, \( S_{\text{min}} = 1 \) creates a highly discriminant tree while \( S_{\text{min}} = 0 \) creates a completely random tree. The main advantage of the random K-D tree compared to other quantization methods is its low computational cost. Quantizing a \( D \)-dimensional descriptor vector using a random K-D tree with \( V \) visual words is \( O(\log_2 V) \). Since a random K-D tree usually has less discriminative power than
other clustering methods, several K-D trees are combined together to obtain a more discriminative codebook. Finally, the resulting BoF histogram is created by concatenating the histograms generated by each K-D tree of the forest.

2.3. Integral Linear Classifier

The integral image representation has been first introduced by Viola and Jones in [24] to quickly extract Haar-wavelet type features. Since then, integral images have been applied to many different tasks like local region descriptors [26], invariant feature extraction [1] or to compute BoF histograms [6]. Inspired by these previous works, we propose to use of integral images to quickly calculate the output score of the linear classifier applied to BoF histograms. If a linear classifier is used to determine the class

![Image with linear classifier weights](image)

Figure 4. Example of the image with the linear classifier weights for each visual words b) obtained from image a).

a $n$-dimensional BoF histogram, then the score of the linear classifier is obtained by the dot product between the BoF histogram $\vec{H}$ and the linear classifier weight vector $\vec{W}$, plus the linear classifier bias factor $b$:

\[
score = \frac{1}{\|\vec{H}\|} \sum_{i=0}^{n} h_i w_i + b \tag{1}
\]

where $h_i$ is the frequency of the $i$-th visual word of the codebook, $\|\vec{H}\|$ is the norm of histogram $\vec{H}$ and $w_i$ is the $i$-th component of the linear classifier weight vector $\vec{W}$. If all components of $\vec{W}$ are positive, then the sum of the previous equation can be calculated using an integral image. Therefore, we define the classifier weight vector $\vec{W}$ components as:

\[
\vec{w}_i = w_i - W_{min} \tag{2}
\]

where $W_{min}$ is the $w_i$ component with the lowest value. Then, replacing $\vec{W}$ by $\vec{w}$ in Eq. 1 the output score of the linear classifier is:

\[
score = \frac{1}{\|\vec{H}\|} \sum_{i=0}^{n} h_i \vec{w}_i + \frac{W_{min}}{\|\vec{H}\|} \sum_{i=0}^{n} h_i + b \tag{3}
\]

When the L1-norm is used to normalize the histogram $\vec{H}$, Eq. 3 can be simplified to:

\[
score = \frac{1}{N} \sum_{i=0}^{n} h_i \vec{w}_i + W_{min} + b \tag{4}
\]

where $N$ is the L1-norm of histogram $\vec{H}$.

Now, an integral image can be used to calculate the sum in Eq. 4: for a linear classifier $c$, let $I_c(x,y)$ be the sum of components $\vec{w}_c$ corresponding to the visual words at pixel coordinates $(x,y)$. In Fig. 4 an example of the $I_c$ image generated from the bike classifier is shown. Then, the integral image $I_c$ can be generated from image $L_c$, so that, the sum of Eq. 4 of a rectangular image region $R$ can be calculated using the integral image $I_c$:

\[
H_R = I_c(x_u, y_u) + I_c(x_b, y_b) - I_c(x_u, y_b) - I_c(x_b, y_u) \tag{5}
\]

where $(x_u, y_u)$ and $(x_b, y_b)$ are respectively the upper left and bottom right corner coordinates of region $R$. Then, the output score of a linear classifier applied to any rectangular image region can be calculated as follows:

\[
score = \frac{1}{N} H_R + W_{min} + b \tag{6}
\]

Using integral images, the computational complexity of classifying any rectangular image region is reduced to 8 memory access, 10 additions and 1 product, independently of the size of rectangular region. Furthermore, when densely sampled features are used, this computational complexity is reduced to 4 memory access, 6 additions and 1 product, as $N$ is known a priori.

2.4. Bag of Features Spatial Bins

Lazebnik et al. show in [12] that adding spatial information using spatial bins in the BoF histograms increases the overall performance of the method. Using spatial bins amounts to computing an independent BoF histogram for every spatial bin and concatenating them to produce the final histogram. In this case, the weights $\vec{w}_c$ of the linear classifier $c$ cannot be accumulated into the same integral image $I_c$, but an integral image has to be calculated for each spatial bin. Therefore, the output score of Eq. 6 when $n$ spatial bins are used is:

\[
score = \sum_{i=0}^{n} \frac{1}{N_i} H_{R_i} + W_{min} + b \tag{7}
\]

where $N_i$ and $H_{R_i}$ are respectively the amount of features and the integral image output in $i$-th spatial bin. The computational complexity of classifying any rectangular image region using $P \times Q$ spatial bins becomes $8 \times P \times Q$ memory access, $8 \times P \times Q + 2$ additions and 1 products, independently of the size of rectangular region. Since we use densely sampled features, the computational complexity can be reduced to $4 \times P \times Q$ memory access, $4 \times P \times Q + 2$ additions and 1 product, as the amount of features $N_i$ at each bin of the rectangular region is known a priori.
3. Pixel level object segmentation

Next we review some of the most relevant contributions to object level pixel segmentation existing in the recent literature. Fulkerson et al. [6] propose a pixel-labeling approach where integral images are effectively used to speed-up the computation of the BoF histograms without relying in suboptimal small dictionaries. Instead, in order to preserve the discriminative power of the codebook while diminishing its size, they define a coarse-to-fine-to-coarse architecture using Agglomerative Information Bottleneck to reduce a full-grown vocabulary tree. In [7] the previous schema is reformulated to use BoF histograms constructed aggregating dense descriptors over a region defined by superpixel neighborhoods instead of square windows. Later a Conditional Random Field (CRF) is used to increase the precision of the method at object boundaries. Although the new schema improves the results, it is not possible to accelerate the accumulation of features in histograms by means of Integral Images as the considered regions are not square, which increases significantly the computational cost of the method.

Marszałek et al. [15] proposed the use of object masks learned from training data to accumulate the evidence provided by visual words computed from sparse local features in consistent object hypothesis, which are subsequently filtered using a discriminative classifier. Shotton et al. [22] use boosting with random feature selection to build a strong classifier using simple texton features and shape masks. A CRF is then used to integrate the different cues (texture, shape, edge, color and location information) and object regions are finally obtained using graph cuts. Pantofaru et al. [20] proposed to combine the output of multiple image segmentation methods to overcome the limitations found by each particular approach. Pixels common to all segmented regions are grouped together assigned a class label that maximizes the marginal probability of each class in the regions that originated the common region. In this work we use this region fusion approach to integrate the results found at different scales as will be seen in next section.

3.1. Object Segmentation

We use the proposed Integral Linear Classifier (ILC) to efficiently assign a category label to each pixel of the image. First, integral features are densely sampled and quantized using a codebook. Then, the integral image of the linear classifier is created from the quantized features. Finally, we calculate the score of the linear classifier for all pixels of the image.

To refine the obtained object segmentation at the boundaries of the objects, the mean-shift is used to extract image regions that have an homogeneous color: we perform mean-shift on a five-dimensional vector composed by LUV color space representation of each pixel and its location in the image. The mean-shift operator is applied using a uniform distribution with a window size proportional to the feature size. Then, the linear classifier score of each pixel of the image is interpreted as \( P(c_{x,y} = k|I) = \varsigma(S(x,y)) \), where \( c_{x,y} \) stands for the class assigned to the pixel \((x,y)\), \( I \) is the image data, \( S_k(x,y) \) is the classifier score for class \( k \), and finally \( \varsigma(\cdot) \) is the sigmoid function that remaps the output of the classifier to the \([0,1]\) interval. These probabilities are accumulated over the regions resulting from the mean-shift segmentation, so that, each region has a certain probability of belonging to an object category. Finally, instead of using this procedure in a single scale as in [6], we repeat this object segmentation procedure at different scales. To combine the resulting segmentations, we use the method proposed by Pantofaru et al. [20]: Pixels which are grouped together by every segmentation should be classified consistently. Thus, the “basic units” of next steps are Intersections of Regions (IofRs). Then, each IofRs is classified by combining the information from all of the individual segments that generated it. Let \( i \) be an IofR, and \( r_i^s \) the region which contains \( i \) in segmentation \( s \). Let \( c_i \) be the class label of \( i \), \( k \) a specific class label, and \( I \) the image data. Then, the segmentation integration method is:

\[
P(c_i = k|I) \propto \sum_s P(c_i^s = k|r_i^s, I)
\]  

(8)

This average over the individual regions’ confidence amounts to marginalizing over the regions containing \( i \), as-
BoF with bins
56
68
Bikes Cars Person
54
Bikes Cars Person
Cascade
65
68
69
Linear
62
51
71.3%
70
55
63
57
55
Cascade
69.6%
64
55
5%
to produce up to
In consequence, we can safely allow the trained classifier
than its negative counterpart (usually the background sam-
responding to pixels of object categories is notably lower
3.2. Cascade classifier
Unfortunately, the amount of positive training data cor-
responding to pixels of object categories is notably lower
than its negative counterpart (usually the background sam-
plies by the ground truth labeled pixels. The computation
times mentioned in the results have been obtained using a
laptop with an Intel 2.6GHz P9600 Core Duo CPU and 4Gb
of RAM.
4.1. Parameter Setup
The results were obtained using the same parameters in
all experiments. Dense features were extracted from square
regions with sizes 20, 28, 40, 56, 78 and 108 pixels, and
gradient was approximated using Haar wavelets of 0.2 times
the considered window size. For SURF and IHOOG descrip-
tors, we used 2 × 2 spatial bins, resulting in 16 and 32
dimensional descriptors respectively. For ISC descriptor, we
used 4 × 4 spatial bins resulting in a 16 dimensional de-
scriptor. The HKM branch factor is set to 10, while for the
ERF we use 10 trees per forest with a randomness factor set
to 0.5. Then, BoF histograms were computed accumulating
the visual words inside a region which doubles the scale of
the feature. Later, those histograms were categorized using
a logistic regression (LR-SVM) support vector machine
[13]. The SVM was trained using all BoF histograms of the
training set, so that, we use a modified version of the LIB-
LINEAR software package [5] that takes advantage of the
high degree of redundancy in BoF histograms to store them
very efficiently in memory.
4.2. GRAZ02 results
For the GRAZ02 dataset, A codebook with 200,000 vi-
sual words in average was obtained for the HKM, while the
codebook size of the obtained ERF was of 500,000 visual
words in average. The cascade classifiers had 2-3 linear
classifiers on average for all object categories. In Table 1
the precision-recall values obtained at equal error rate for
the different configurations of the method are shown. Spa-
tial bins in the BoF histogram had little effect in the per-
formance of the method. However, the cascade classifier
together with an ERF codebook increased the performance
of the method by 2.6%, 2.4% and 13.12% for bikes, cars

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Codebook</th>
<th>BoF without spatial bins</th>
<th>BoF with bins</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linear</td>
<td>Cascade</td>
<td>Linear</td>
</tr>
<tr>
<td>SURF HKM</td>
<td>Bikes</td>
<td>Person</td>
<td>Bikes</td>
</tr>
<tr>
<td></td>
<td>61.3%</td>
<td>58.6% 51.8%</td>
<td>64.3% 61.7%</td>
</tr>
<tr>
<td></td>
<td>Cars</td>
<td>Person</td>
<td>Cascade</td>
</tr>
<tr>
<td></td>
<td>51.6%</td>
<td>58.6% 41.2%</td>
<td>Cascade</td>
</tr>
<tr>
<td></td>
<td>Person</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>58.1%</td>
<td>51.6% 46.1%</td>
<td>Cascade</td>
</tr>
<tr>
<td>IHOOG HKM</td>
<td>Bikes</td>
<td>Person</td>
<td>Bikes</td>
</tr>
<tr>
<td></td>
<td>68.1%</td>
<td>54.9% 51.5%</td>
<td>69.9% 57.9%</td>
</tr>
<tr>
<td></td>
<td>Cars</td>
<td>Person</td>
<td>Cascade</td>
</tr>
<tr>
<td></td>
<td>51.6%</td>
<td>58.6% 41.2%</td>
<td>Cascade</td>
</tr>
<tr>
<td></td>
<td>Person</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>58.1%</td>
<td>51.6% 46.1%</td>
<td>Cascade</td>
</tr>
<tr>
<td>ISC HKM</td>
<td>Bikes</td>
<td>Person</td>
<td>Bikes</td>
</tr>
<tr>
<td></td>
<td>51.9%</td>
<td>41.2% 48.1%</td>
<td>53.9% 46.6%</td>
</tr>
<tr>
<td></td>
<td>Cars</td>
<td>Person</td>
<td>Cascade</td>
</tr>
<tr>
<td></td>
<td>51.6%</td>
<td>58.6% 41.2%</td>
<td>Cascade</td>
</tr>
<tr>
<td></td>
<td>Person</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>58.1%</td>
<td>51.6% 46.1%</td>
<td>Cascade</td>
</tr>
</tbody>
</table>

Table 1. Precision-Recall values at equal error rate obtained in the GRAZ02 dataset for each parameter combination.
4.4. Computational Cost Evaluation

Regarding the computational cost of the proposed approach, the average time needed to construct the codebook is of 102 seconds using HKM and 250 seconds with ERF. The time required to train a linear classifier for an object category using LR-SVM is about 3 minutes for HKM and 17 minutes using ERF. When a cascade classifier is used the time needed to train increases to 6-8 minutes for the HKM and to 25-40 minutes for the ERF, depending on the number of classifiers trained in the cascade. The time required to extract and quantize densely sampled features from the GRAZ02 dataset images.

Table 3. Average time required to extract and quantize densely sampled features in the GRAZ02 dataset images.

<table>
<thead>
<tr>
<th>Codebook</th>
<th>SURF</th>
<th>HOG</th>
<th>ISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKM</td>
<td>181.7ms</td>
<td>298.5ms</td>
<td>170.9ms</td>
</tr>
<tr>
<td>ERF</td>
<td>175.7ms</td>
<td>293.5ms</td>
<td>166.7ms</td>
</tr>
</tbody>
</table>

Table 2. Accuracy results obtained in the VOC2007 dataset. In **Codebook**, H stands for HKM and E stands for ERF. In **Separations**, Y means spatial bins, while N means the contrary. Finally, in **Classifier**, L and C stand for linear and cascade classifier respectively.
count that their time results are obtained with images from the MSRC database (300 × 213 pixel images), while ours are of GRAZ02 database (640 × 480 pixel images): the images used in our evaluation are 4.5 times bigger, thus the difference in computation time. When evaluated in the MSRC database the time per image needed by our method to obtain the 21-class segmentation result was of 91ms (11 fps).

5. Conclusions

In this paper we present an efficient method to obtain a multiclass pixel-level segmentation of an image in less than 0.5 seconds. Our main contribution is the introduction of the Integral Linear Classifier, which is used to bypass the accumulation step in a Bag of Features schema and directly obtain the classification score for an arbitrary sub-window of the image. In our experiments, we show that our method obtains results comparable to the state of the art in two challenging datasets but with a much lower computational cost. Besides, we have compared the performance of three efficient feature descriptors (SURF, IHOG and ISC) and two different codebooks (HKM and ERF). Results show that the SURF descriptor gives a good compromise between speed and accuracy, the IHOG obtains better results at the price of a higher computational cost, and the ISC descriptor is the fastest to compute but has modest results. Regarding the quantizers, the ERF has shown to outperform both in accuracy and speed the HKM. Finally, we have found little improvement in the use of spatial bins in the BoF, while the cascade classifier consistently improved the results by handling the asymmetry between false positives and false negatives.

Acknowledgements

This work was supported by the Spanish Ministry of Education projects TIN 2006-15308-C02-02, MIPRCV Consolider Ingenio 2010 and the grant 2009-SGR-1434 of the Generalitat de Catalunya.

References

[17] D. Nister and H. Stewenius. Scalable recognition with a good compromise between speed and accuracy, the IHO G obtains better results at the price of a higher computational cost, and the ISC descriptor is the fastest to compute but has modest results. Regarding the quantizers, the ERF has shown to outperform both in accuracy and speed the HKM. Finally, we have found little improvement in the use of spatial bins in the BoF, while the cascade classifier consistently improved the results by handling the asymmetry between false positives and false negatives.

Acknowledgements

This work was supported by the Spanish Ministry of Education projects TIN 2006-15308-C02-02, MIPRCV Consolider Ingenio 2010 and the grant 2009-SGR-1434 of the Generalitat de Catalunya.

References

[17] D. Nister and H. Stewenius. Scalable recognition with a good compromise between speed and accuracy, the IHO G obtains better results at the price of a higher computational cost, and the ISC descriptor is the fastest to compute but has modest results. Regarding the quantizers, the ERF has shown to outperform both in accuracy and speed the HKM. Finally, we have found little improvement in the use of spatial bins in the BoF, while the cascade classifier consistently improved the results by handling the asymmetry between false positives and false negatives.

Acknowledgements

This work was supported by the Spanish Ministry of Education projects TIN 2006-15308-C02-02, MIPRCV Consolider Ingenio 2010 and the grant 2009-SGR-1434 of the Generalitat de Catalunya.

References