

# Localizing Objects with Smart Dictionaries

Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto

Department of Computer Science,  
University of California, Los Angeles, CA 90095  
{bfulkers,vedaldi,soatto}@cs.ucla.edu

**Abstract.** We present an approach to determine the category and location of objects in images. It performs very fast categorization of each pixel in an image, a brute-force approach made feasible by three key developments: First, our method reduces the size of a large generic dictionary (on the order of ten thousand words) to the low hundreds while increasing classification performance compared to  $k$ -means. This is achieved by creating a discriminative dictionary tailored to the task by following the information bottleneck principle. Second, we perform feature-based categorization efficiently on a dense grid by extending the concept of integral images to the computation of local histograms. Third, we compute SIFT descriptors densely in linear time. We compare our method to the state of the art and find that it excels in accuracy and simplicity, performing better while assuming less.

## 1 Introduction

Bag-of-features methods have enjoyed great popularity in object categorization, owing their success to their simplicity and to surprisingly good performance compared to more sophisticated models and algorithms. Unfortunately, such methods only provide an answer as to whether an image contains an object of a certain category, but they do not offer much insight as to where that object might be within the image. In other words, because the representation discards spatial information, bag-of-features methods cannot be used for localization directly.

That is, unless one could devise an object categorization method efficient enough to test at a window centered on each pixel of an image. In that case, one would be able to exploit the co-occurrence of features within a local region and localize the object, pixel by pixel. However, with many features detected in each image and quantized into thousands or tens of thousands of “words,” this does not appear to be a viable proposition, especially in light of recent results that advocate using very large visual dictionaries [1,2,3].

But what if we could reduce the size of a dictionary from tens of thousands of words to a few hundred and maintain improved localization? After all, dictionaries commonly used in bag-of-features are not designed for the specific task of categorization, so there may be gains to be found in creating “smarter” dictionaries that are tailored to the task. This is precisely what we set out to do. With this we can obtain robust, efficient localization, and show that our scheme



**Fig. 1.** Upper Left: Original image. Middle: Labeling weighted by the confidence for the class "person". Lower Left: Labeling weighted by the confidence, with low confidence background pixels reclassified as foreground. Right: Labeling weighted by the confidence, with low confidence foreground pixels reclassified as background.

performs better than the state of the art [4] on a challenging dataset [5] despite its simplicity.

**Contributions.** In this paper we propose a method for pixel-level category recognition and localization. We employ a simple representation (bag-of-features) and contribute three techniques which make the categorization process efficient. First, we extend integral images [6] to windowed histogram-based classification. Second, we construct small dictionaries which maintain the performance of their larger counterparts by using agglomerative information bottleneck (AIB) [7]. In order to greatly reduce the bottleneck of quantizing features, we construct the large dictionaries using hierarchical  $k$ -means (HKM). We also propose an important speedup which makes it possible to compute AIB on large dictionaries with ease. Third, we show that we can compute SIFT features densely in linear time.

**Related work.** Lazebnik *et al.* [8] also perform discriminative learning to optimize  $k$ -means, but are limited to small dictionaries and visual words which are Voronoi cells. Leibe *et al.* [9] also perform compression, but not in a discriminative sense. Finally, Winn *et al.* [10] do discriminative compression in a similar fashion, but we show that we perform better and can scale to larger dictionaries. For the task of pixel-level localization, we show that our method outperforms  $k$ -means and Winn *et al.*, while being nearly two hundred times faster to construct. We compare our method directly to Winn *et al.* [10] in Sect. 3.1.

Object categorization methods have matured greatly in recent years, going beyond bags of features by incorporating a spatial component into their model. Approaches are varied, but broadly tend to include one of the following: interactions between pairs of features [11,12,13], absolute position of the features [14], segmentation [15,16], or a learned shape or parts model of the objects [4,17,18]. Our method exploits interaction between groups of features (all features in the

window), but does not explicitly represent their configuration, in the spirit of achieving viewpoint-invariance for objects of general shape[19].

Regarding object localization, recent works are based on two different approaches: either they form a shape based model of the object class as in [4,17], or they enforce spatial consistency using a conditional random field (CRF) [20,21]. We focus our comparisons on the method of Marszalek *et al.* [4], who forms a family of shape models for each category from the training data and casts these into the target image on sparse feature points to find local features which agree on the deformation of one of the learned models. Our approach will show that we obtain better performance by simply performing local classification at every pixel.

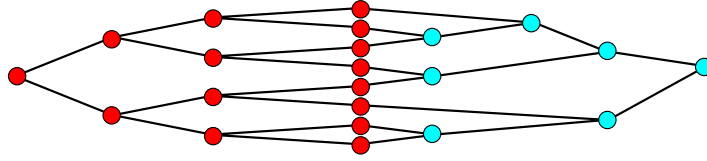
Along the way, we will construct a small, smart dictionary which is comprised of clusters of features from a much larger dictionary using AIB [7]. Liu *et al.* [22] recently proposed a co-clustering scheme maximizing mutual information (MMI) for scene recognition. Agarwal *et al.* [23] cluster features to create a whole image descriptor called a “hyperfeature” stack. Their scheme repeatedly quantizes the data in a fixed pyramid, while our representation allows the computation of any arbitrary window without incurring any additional computational penalty. We can just as easily extract our bag-of-features for the whole image, blocks of the image, or (as we show) each pixel on a grid.

After this paper was submitted, two additional related works were published. Lampert *et al.* [24] use branch-and-bound to search all possible subwindows of an image for the window which best localizes an object. They do not seek to localize at the pixel level, or handle multiple objects in one image. Shotton *et al.* [25] perform pixel labeling as we do, but use much simpler features combined with randomized decision forests. Because they use simple features, they must build their viewpoint invariance by synthetically warping the training images and providing them as new training examples. Our framework allows for that, but our descriptors already exhibit reduced sensitivity to viewpoint.

## 2 Brute-Force Localization

Our method uses bag-of-features as a “black box” to perform pixel-level category recognition. In the simplest case, this involves extracting features from the image and then for each window aggregating them into a histogram and comparing this histogram with the training data we have been provided. The main stumbling block is the extraction of histograms at each pixel of the image. For this, we use integral images. However, this alone is not sufficient: Using large dictionaries in the setting we propose would be impossible, yet we need our dictionary to remain discriminative in order to be useful. To this end, in Sect. 3 we propose a method for building a compact, efficient and informative dictionary from a much larger one.

**Integral Images.** Viola *et al.* [6] popularized the use of integral images for the task of feature extraction in boosting, and it has since been used by others [20]



**Fig. 2. Dictionary architecture.** We use hierarchical  $k$ -means (HKM) to build a vocabulary tree (left, red nodes) of finely quantized features by recursively partitioning the data. Next, we use AIB to build an agglomerative tree (right, blue nodes) of informative words. This architecture is efficient (in training and testing) and powerful.

for similar purposes. Integral images can also be used to quickly count events in image regions [26], and Porikli [27] shows how to compute integral histograms in Cartesian spaces. We build integral images of spatial occurrences of features and use them to efficiently extract histograms of visual words on arbitrary portions of the image. For each visual word  $b$  in our dictionary, let  $O_b(x, y)$  be the number of occurrences of  $b$  at pixel  $(x, y)$  (typically this number is either zero or one). Each image  $O_b$  is transformed into a corresponding integral image  $I_b$  by summing over all the pixels  $(x', y') \leq (x, y)$  above and to the left of pixel  $(x, y)$ :

$$I_b(x, y) = \sum_{x' < x} \sum_{y' < y} O_b(x', y')$$

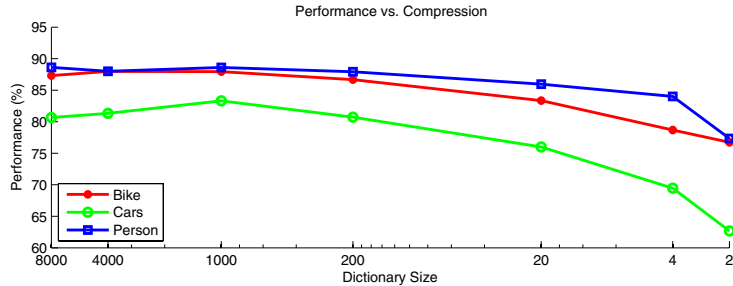
Let  $R$  be a rectangular image region. The histogram  $H_R(b)$  is the number of occurrences of  $b$  in  $R$  and can be quickly computed as:

$$H_R(b) = I_b(x_s, y_s) + I_b(x_e, y_e) - I_b(x_s, y_e) - I_b(x_e, y_s)$$

where  $(x_s, y_s)$  is the upper left corner and  $(x_e, y_e)$  is the lower right corner of  $R$ . In this way we can extract a histogram of feature occurrences for a window of arbitrary size in constant time. The memory required scales with the size of the image and the size of the dictionary, and the constant time required to construct each histogram scales linearly with the size of the dictionary. This precludes the use of very large dictionaries, because each dictionary element that is included requires adding an integral image.

### 3 Informative, Compact and Efficient Dictionaries

Our localization method directly benefits from having a small dictionary because the complexity is linear in its size. Yet many recent works [1,2,3,29] indicate that large or very large dictionaries perform better for both object recognition and categorization. However, over-specific visual words should eventually over-fit the data, especially in categorization. We argue that one of the reasons why large dictionaries often outperform smaller ones is that dictionaries are usually not optimized for discrimination. If visual words could be tailored to discriminate



**Fig. 3.** Results of an experiment showing the performance of AIB as the dictionary is compressed. We adopt the framework of [28] on Graz-02, extracting SIFT descriptors on salient regions, quantizing them, and classifying the resulting histograms with an SVM. We vary the compression of the dictionary, starting from the full HKM tree (8,000 leaves,  $K=20$ ) and compressing to a dictionary with only 2 elements. In each case, we can compress the dictionary by a factor of 8 without losing any accuracy. In some cases (Cars, Bikes) we even increase performance slightly.

different categories, a smaller number of them would be sufficient. Motivated by this idea, we seek to gain the performance increases of recent approaches using large dictionaries without their computational burden.

Winn *et al.* [10] introduced the idea of constructing small and informative visual dictionaries by compressing larger ones. Here we propose a novel architecture and compression algorithm that has two key advantages: (i) it is very fast to project novel features on the optimized dictionary and (ii) compression is several orders of magnitude faster, which makes it possible to operate on much larger dictionaries and datasets. In addition, we show that our method outperforms [10] for the task of pixel level categorization (Sect. 4).

**Fast Projection by HKM.** In order to project  $N$  novel features  $f \in \mathcal{F} \subset \mathbb{R}^n$  onto a visual dictionary of  $L$  elements, the required time is usually  $O(NL)$ . This is true even if the dictionary is eventually compressed into a smaller one [10]. Since a large number of features  $N$  are typically extracted from an image, mapping features to the visual dictionary may become the bottleneck of the recognition pipeline.

Here we solve this problem by using an HKM [1] tree as the initial visual dictionary. HKM trees have shown excellent performance in object recognition [1,2]. More importantly, they enable efficient projection of novel features, requiring only  $O(N \log L)$  operations. Combining the HKM tree with the compression tree (Sect. 3.1), yields the coarse-to-fine-to-coarse architecture of Fig. 2.

### 3.1 Dictionary Compression

We compress a visual dictionary by merging visual words in such a way that the discriminative power of the dictionary is preserved. The discriminative power can be characterized in different ways, yielding different compression algorithms. Here we

discuss and compare two: Agglomerative Information Bottleneck (AIB) [7] and the method from [10], which we indicate with WCM. We also contribute a modification of the AIB algorithm that makes it feasible to process dictionaries of tens of thousands of elements. We show that the same fast algorithm may be used to speed-up WCM as well. However, even with this speedup we find that WCM is much slower than AIB (to the point of being infeasible for large datasets and dictionaries) and performs worse than AIB when applied to pixel-level categorization.

**AIB Compression.** AIB characterizes the discriminative power of the dictionary  $\mathcal{X}$  as the mutual information  $I(x, c)$  of the random variables  $x$  (visual word) and  $c$  (category):

$$I(x, c) = \sum_{x \in \mathcal{X}} \sum_{c=1}^C P(x, c) \log \frac{P(x, c)}{P(x)P(c)}. \quad (1)$$

The joint probability  $P(x, c)$  is estimated from data simply by counting the number of occurrences of each visual word  $x \in \mathcal{X}$  in each category  $c \in \{1, \dots, C\}$ . AIB iteratively compresses the dictionary  $\mathcal{X}$  by merging the two visual words  $x_i$  and  $x_j$  that cause the smallest decrease  $D_{ij}$  in the mutual information (discriminative power)  $I(x, c)$ . Denoting  $[x]_{ij}$  the random variable corresponding to the dictionary after the merge, the quantity  $D_{ij}$  is

$$D_{ij} = I(x, c) - I([x]_{ij}, c). \quad (2)$$

The information  $I(x, c)$  is monotonically reduced after each merge. Merging is iterated until one obtains the desired number of words.

At test time, projecting a visual word  $x \in \mathcal{X}$  onto the compressed dictionary requires constant time ( $O(1)$ ). So, since we use HKM for the initial dictionary, the number of operations required to project  $N$  novel features on the compressed dictionary is only  $O(N \log K)$ , where  $K$  is the number of leaves of the HKM tree.

In Fig. 3 we show the effectiveness of this technique using a simple experiment on Graz-02. In all cases, we compress the dictionary significantly without losing any accuracy. In fact, in two of the three cases the results are slightly improved at some compression level.

**Fast AIB.** The basic implementation of the AIB algorithm is prohibitively slow for very large dictionaries. The implementation proposed in Slonim *et al.* [7] stores the symmetric “distance” matrix  $D = [D_{ij}]$  ( $O(L^2)$  space).<sup>1</sup>

Then, at each iteration one only needs to update the row and column  $i, j$  of  $D$  which were involved in the last merge (since only words  $x_i$  and  $x_j$  change).

<sup>1</sup> **Reciprocal Nearest Neighbor Clustering** [9] proposes an efficient agglomerative clustering algorithm that can be applied whenever the distance matrix  $D_{ij}$  satisfies the *reducibility property*  $D_{ij} \leq \min\{D_{ik}, D_{jk}\} \Rightarrow \min\{D_{ik}, D_{jk}\} \leq D_{\bar{i}\bar{j},k}$ , where  $\bar{i}\bar{j}$  denotes the merged dictionary entry. Unfortunately, AIB clustering violates this property. For a counter example, consider the case  $C = 3$ ,  $P(x_i) = P(x_j) = P(x_k) = 1/3$ ,  $P(c = 1|x_k) = P(c = 2|x_k) = 1/3$ ,  $P(c = 1|x_i) = P(c = 2|x_i) = 2/5$  and  $P(c = 2|x_j) = P(c = 3|x_j) = 2/5$ .

This has complexity  $O(LC)$ . Searching for the minimal matrix element at each step is  $O(L^2)$ , and this process is iterated  $L$  times, so the overall complexity is  $O(L(L^2 + LC))$  time and  $O(L^2)$  space [30].

A simple modification of the basic algorithm is far more efficient. We cache for each  $i$  the index and value  $(k_i, D_{ik_i})$  of the minimum distance along the row and do not store  $D$ . This reduces the time spent searching for the minimum element  $(i^*, j^*)$  of  $D$  from  $O(L^2)$  to  $O(L)$ . Now, when we merge  $(i^*, j^*)$ , we must update the entries  $(k_i, D_{ik_i})$  for which either  $k_i = i^*$  or  $k_i = j^*$ . This has time complexity  $O(L(L + \gamma LC))$ , where  $\gamma$  is the number of entries which need to be updated at each iteration. We find empirically that  $\gamma \ll L$ , so in practice the amount of time taken is approximately  $O(L^2C)$  and the space complexity has been reduced to  $O(L)$ .

To get a sense of the advantages of this implementation, the original AIB algorithm [30] requires  $L^2$  elements of memory at each iteration, which meant that a 20,000 cluster case would require roughly 3.2GB of memory as opposed to 320kB with our modified approach. We also note that in the 10,000 cluster cases we test, we often find  $\gamma$  to be on the order of 5 and so the clustering process is very fast (about 5 minutes for 10,000 clusters on a 2.3Ghz Core 2 Duo). The basic implementation of AIB on the same task requires approximately a day.

**WCM compression.** WCM differs from AIB in the way it measures the discriminative power of the visual dictionary. This is motivated by the fact that in the bag-of-features setting images are represented by histograms of visual words rather than visual words in isolation. Thus, one is more interested in obtaining *informative histograms* than informative visual words. This notion could be captured, for instance, by considering the mutual information  $I(h, c)$  in place of the information  $I(x, c)$  used by AIB.

Due to the high dimensionality of the histograms, estimating  $I(h, c)$  is nearly impossible without strong assumptions. WCM assumes that histograms are distributed according to a mixture of Gaussians, with one Gaussian per category. Moreover, they characterize the discriminative power of the dictionary by the category posterior probability  $p(c|h)$  rather than by the information  $I(h, c)$ . This creates a mechanism for model selection which can automatically stop the merging procedure when a maximum of  $p(c|h)$  is attained (in contrast, in AIB the information criterion  $I(x, c)$  decreases monotonically). Finally, it is also possible to extend the fast AIB algorithm introduced in the previous section to WCM almost without changes.

Despite these appealing characteristics, WCM does not perform as well as AIB in our setting. First, despite our fast implementation, it is much slower than AIB on large datasets (in Sect. 4 we show it requires up to twelve days on a task that our fast AIB can solve in about five minutes).<sup>2</sup> Second, WCM model selection is not useful for our localization task as we are interested in obtaining

<sup>2</sup> Updating an entry of the  $D_{ij}$  matrix requires scanning the data to compute the linear correlation of bin  $i$  and  $j$ . This is due to the fact that WCM considers visual words in the context of histograms where AIB does not. Although the model assumes that histogram bins are statistically independent, they interact when merged. The update operation requires about  $O(ML^2C)$ , where  $M$  is the number of training histograms, as opposed to  $O(L^2C)$  for AIB.

dictionaries of a prescribed size (Sect. 4). Third, AIB compressed dictionaries result in better categorization results than WCM<sup>3</sup> (Sect. 4; Table 1).

## 4 Experiments

Graz-02 [5] is a challenging dataset consisting of three categories (cars, bicycles, and people) with extreme variability in pose, scale and lighting. Our goal is the same as Marszalek *et al.* [4]: We wish to label each image pixel as either belonging to one of these categories or not. In order to compare directly to Marszalek *et al.* [4], we adopt their measure of performance: pixel precision-recall. Our features extraction and dictionary compression are implemented within VLFeat [31], and the rest of our implementation is available from our website<sup>4</sup>.

**Training.** We select the same training images as [4], namely the first 150 odd numbered images from each category. We compute dense SIFT descriptors and quantize them using our dictionary (see Sect. 4). Then for each image we generate two histograms: The first aggregates all the features that belong to the background (based on the feature center and the ground truth object masks), and the second the features that belong to the object. This collection of histograms is used as training data for either an SVM classifier with  $\chi^2$  kernel or an inverse document frequency (IDF) [1] weighed  $k$ -nearest neighbor (KNN) classifier ( $k = 10$ ).

**Fast Dense Feature Extraction.** We extract a SIFT descriptor [32] every four pixels. The support of each descriptor is a  $16 \times 16$  patch. We do not compute the orientation of the descriptor since this has been shown to adversely affect other dense bag of features methods [28]. Features that have low gradient magnitude before normalization are discarded as in [14,3].

We introduce here a novel technique to compute dense SIFT descriptors very efficiently. Fast SIFT-like descriptors have been proposed by [33,3] and recently [34]. Our technique has the advantage of being fully equivalent to SIFT and still efficient: The complexity is only  $O(Q^2R)$  compared to  $O(Q^2R^2)$  of a direct implementation, where  $Q^2$  the area of the image and  $R^2$  the area of the descriptor support. Moreover, up to a small approximation, we can reduce the complexity to  $O(Q^2)$ , which is independent of the area of the descriptor support. Our implementation is included with VLFeat [31], an open source feature extraction library.

The idea is to reduce the calculation of the dense descriptors to a number of separable convolutions. Recall that the SIFT descriptor at location  $(x_0, y_0)$  is a three-dimensional histogram of the gradient  $\nabla I(x, y)$  in a circular patch surrounding that point [32]. The histogram is indexed by the relative position  $(x - x_0, y - y_0)$  and orientation  $\angle \nabla I(x, y)$  of the gradient  $\nabla I(x, y)$  in the patch, weighed by the gradient modulus  $|\nabla I(x, y)|$  and by a Gaussian window

<sup>3</sup> This is probably due to the fact that in our setting the assumptions made by WCM are not satisfied.

<sup>4</sup> <http://vision.ucla.edu/bag/>

centered at  $(x_0, y_0)$ . The relative positions are quantized in  $4 \times 4$  bins and the orientation in 8 bins using bilinear interpolation. For a given orientation, the data for a bin  $b$  is obtained by computing integrals of the type  $\int g(x - x_0, y - y_0)h_b(x - x_0, y - y_0)f(x, y) dx dy$ , where  $f(x, y)$  is the mass of the gradient at that particular orientation,  $g(x, y)$  is the Gaussian window and  $h_b(x, y)$  is the product of two triangular windows resulting from the bilinear interpolation of bin  $b$ . Since both  $h(x, y)$  and  $g(x, y)$  are separable, the calculation requires only  $O(Q^2R)$  operations.

Notice that this requires  $4 \times 4 \times 8$  separable convolutions in total. However, by dropping the Gaussian window  $g(x, y)$  (the effect on the computed descriptors is modest), convolutions for different spatial bins at the same orientations are identical up to translation, and only 8 separable convolutions are sufficient. Moreover, recall that convolving by a rectangular kernel can be done very efficiently by integral images. Since convolving by a triangular kernel can be decomposed in convolving twice by rectangular ones, we obtain a final complexity of  $O(Q^2)$ .

We also experimented with color descriptors by first transforming the  $(R, G, B)$  image into the normalized  $(r, g, b)$  space [35] where  $r = \frac{R}{R+G+B}$ ,  $g = \frac{G}{R+G+B}$ ,  $b = \frac{B}{R+G+B}$ . SIFT descriptors are extracted independently from the  $r$  and  $g$  channel and concatenated into one 256 dimensional descriptor<sup>5</sup>.

**Dictionary Construction.** We sample a large number of feature-category pairs from our training data and follow one of two approaches to construct a dictionary. As a baseline, we use  $k$ -means with  $k = \{5, 40, 200\}$ . Alternatively, we construct a hierarchical  $k$ -means dictionary with  $k = 10$  and 10,000 leaf nodes, and then compress this dictionary to  $N = \{5, 40, 200\}$  clusters (we experiment with both AIB and WCM). Notice that, in our application, the size of the dictionary is the primary factor in determining the speed and memory footprint of the classification algorithm.

**Testing.** We test on the first 150 even numbered images from each category. For each pixel on a grid with a step 4 pixels, we construct a histogram of feature occurrences within a window of  $80 \times 80$  pixels using integral images (Sect. 2) and classify using either SVM or KNN. The classification returns a label and a score. The magnitude of the score indicates the confidence in the label and the sign of the score indicates the class (-1 is a fully confident classification of “background”). For pixels which do not lie on the grid, we interpolate the score from adjacent pixels.

We choose a range of confidence thresholds  $\rho$  and for each we classify as object all pixels which have a score greater than the threshold. These are compared to the ground-truth segmentation which provides us with pixel precision and recall for the testing data. We also use this threshold to create Fig. 1 and to generate the movie included as supplementary material.

---

<sup>5</sup> We do not include the  $b$  channel because the constraint  $r + g + b = 1$  makes it redundant.

**Table 1.** A comparison of the pixel precision-recall equal error rates on Graz-02. Although we do not represent shape explicitly, our results are competitive with [4]. The best performance is achieved using our compressed dictionary (Sect. 3). We also outperform Winn *et al.* (WCM), and while our dictionaries take roughly 5 minutes to construct, Winn *et al.* takes up to 12 days on this task. Here time is the amount of time required per image, including dense feature extraction, quantization, and classification of all pixels. Dense feature extraction alone requires 0.15s for grayscale and 0.3s for RGB. Images are  $640 \times 480$ .

object class	cars	people	bicycles	time
[4] no hyp. eval.	40.4%	28.4%	46.6%	-
[4] no evid. collect.	50.3%	40.3%	48.9%	-
[4] full framework	53.8%	44.1%	61.8%	-
AIB5-KNN	39.8%	47.1%	57.4%	0.5s
AIB5-SVM	38.5%	48.2%	56.8%	0.7s
KM5-KNN	27.1%	32.1%	44.9%	2s
KM5-SVM	30.0%	33.1%	44.9%	2s
AIB40-KNN	47.5%	47.2%	61.7%	0.5s
AIB40-SVM	44.9%	49.0%	59.9%	0.8s
KM40-KNN	45.1%	42.8%	59.5%	2s
KM40-SVM	37.8%	45.4%	59.5%	2.5s
AIB200-KNN	50.9%	49.7%	63.8%	1.1s
AIB200-SVM	40.1%	50.7%	59.9%	3.3s
KM200-KNN	50.1%	46.5%	62.6%	2.5s
KM200-SVM	39.3%	49.3%	58.9%	5s
AIB200RGB-KNN	<b>54.7%</b>	47.1%	<b>66.4%</b>	1.4s
AIB200RGB-SVM	49.4%	<b>51.4%</b>	65.2%	3.7s
WCM200RGB-KNN	54.2%	41.1%	59.6%	1.4s
WCM200RGB-SVM	39.8%	46.3%	59.6%	3.7s
KM200RGB-KNN	51.6%	44.2%	60.8%	3.5s
KM200RGB-SVM	48.3%	49.3%	61.4%	7s

**Discussion.** Table 1 reports the points where precision and recall are equal and compares our results to those of Marszalek *et al.* [4], the previous state of the art in pixel accurate localization on Graz-02. The full curves are available at the author’s website. Although we do not have shape or even scale in our model, we still perform significantly better on all categories. Specifically, our best performing cases are 4.8% better on bikes, 0.9% better on cars, and 7.3% better on people. In each case, the compressed dictionary outperforms the  $k$ -means dictionary of equal size. The differences decrease as the final vocabulary size is increased, which is intuitive because the variability of the dataset can be better captured by  $k$ -means as we increase  $k$ , while the descriptive power of our rebuilt dictionary is upper bounded by that of the associated HKM tree.

Our approach naturally provides a confidence measure, so we can quantify the uncertainty in classification as shown in Fig. 4.



**Fig. 4. Selected results on Graz-02.** (Best viewed in color). Images are first masked by the classification then transformed to HSV. The HSV images have their V channel weighted by the confidence in the classification, darkening the pixels which are less confident about the class. All images shown were generated with the parameter set denoted AIB200RGB and classified with an SVM.

## 5 Conclusions and Future Work

We have described and shown that an object localization framework which uses bag-of-features as a tool can successfully localize objects without making assumptions about the shape of the object, or explicitly performing segmentation.

In order to make this possible, we have also shown a method that efficiently learns a dictionary which is tailored for the task of categorization. In spite of its simplicity, our approach produces pixel-accurate object localizations which exceed the state of the art on a challenging dataset.

Our experiments show that more care should be exercised in integrating shape information into generic object class representations. We believe shape is an important discriminant ([19], Theorem 3), but our work should be viewed as a baseline method whose performance should be convincingly exceeded before justifying the additional complexity a shape-based model might bring.

The techniques we describe can be directly extended to pixel-level multi-class localization, and we plan to do this. We will also explore adding a notion of scale, perhaps by simply performing multiple classifications at different scales followed by scale selection. We note that in our framework this does not add any significant computational burden since our complexity is not tied to the size of the windows we choose. Our system is already very fast, and we plan to improve the speed further until the system operates in real-time.

Last, our approach could be combined with conditional random fields or other models that are capable of enforcing spatial consistency and context-type constraints (e.g. [20,16]). However, we note that we already have some local consistency built-in since each windowed histogram we classify has a very high overlap with its neighbors.

## Acknowledgements

This research was supported by ONR N00014-08-1-0414, 67F-1080868 and AFOSR FA9550-06-1-0138.

## References

1. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Proc. CVPR (2006)
2. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: Proc. CVPR (2007)
3. Tuytelaars, T., Schmid, C.: Vector quantizing feature space with a regular lattice. In: Proc. ICCV (2007)
4. Marszałek, M., Schmid, C.: Accurate object localization with shape masks. In: Proc. CVPR (2007)
5. Opelt, A., Pinz, A.: Object localization with boosting and weak supervision for generic object recognition. In: Kalviainen, H., Parkkinen, J., Kaarna, A. (eds.) SCIA 2005. LNCS, vol. 3540, pp. 862–871. Springer, Heidelberg (2005)
6. Viola, P., Jones, M.: Robust real-time object detection. In: Second International Workshop on Statistical and Computational Theories of Vision, Vancouver, Canada (2001)
7. Slonim, N., Tishby, N.: Agglomerative information bottleneck. In: Proc. NIPS (1999)

8. Lazebnik, S., Raginsky, M.: Learning nearest-neighbor quantizers from labeled data by information loss minimization. In: Proc. Conf. on Artificial Intelligence and Statistics (2007)
9. Leibe, B., Micolajczyk, K., Schiele, B.: Efficient clustering and matching for object class recognition. In: Proc. BMVC (2006)
10. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: Proc. ICCV (2005)
11. Marszałek, M., Schmid, C.: Spatial weighting for bag-of-features. In: Proc. CVPR (2006)
12. Leordeanu, M., Hebert, M., Sukthankar, R.: Beyond local appearance: Category recognition from pairwise interactions of simple features. In: Proc. CVPR (2007)
13. Ling, H., Soatto, S.: Proximity distribution kernels for geometric context in category recognition. In: Proc. CVPR (2007)
14. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bag of features: Spatial pyramid matching for recognizing natural scene categories. In: Proc. CVPR (2006)
15. Cao, L., Fei-Fei, L.: Spatially coherent latent topic model for concurrent object segmentation and classification. In: Proc. ICCV (2007)
16. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: Proc. ICCV (2007)
17. Leibe, B., Leonardis, A., Schiele, B.: Combined object categorization and segmentation with implicit shape model. In: ECCV Workshop on Statistical Learning in Comp. Vision (2004)
18. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multi-scale, deformable part model (2007), <http://people.cs.uchicago.edu/pff/papers/>
19. Vedaldi, A., Soatto, S.: Features for recognition: Viewpoint invariance for non-planar scenes. In: Proc. ICCV (2005)
20. Shotton, J., Winn, J., Rother, C., Criminisi, A.: *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: Proc. ECCV (2006)
21. He, X., Zemel, R., nán, M.C.P.: Multiscale conditional random fields for image labeling. In: Proc. CVPR (2004)
22. Liu, J., Shah, M.: Scene modeling using co-clustering. In: Proc. ICCV (2007)
23. Agarwal, A., Triggs, B.: Hyperfeatures - multilevel local coding for visual recognition. Technical report, INRIA (2005)
24. Lampert, C., Blaschko, M., Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. *cvpr* (2008)
25. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: CVPR (2008)
26. Wang, X., Doretto, G., Sebastian, T., Rittscher, J., Tu, P.: Shape and appearance context modeling. In: Proc. ICCV (2007)
27. Porikli, F.: Integral histogram: A fast way to extract histograms in cartesian spaces. In: Proc. CVPR (2005)
28. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV* (2006)
29. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: Proc. NIPS (2006)
30. Slonim, N.: Iba.1.0: Matlab code for information bottleneck clustering algorithms (2003), <http://www.princeton.edu/nslonim/>
31. Vedaldi, A., Fulkerson, B.: Vlfeat: Feature extraction library (2007), <http://vision.ucla.edu/vlfeat/>

32. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *IJCV* 2(60), 91–110 (2004)
33. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. In: Proc. ECCV (2006)
34. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: Proc. CVPR (2008)
35. Elgammal, A., Harwood, D., Davis, L.: Non-parametric model for background subtraction. In: Proc. ECCV, pp. 751–767 (2000)