# Vector Quantizing Feature Space with a Regular Lattice

Tinne Tuytelaars [*]
K.U.Leuven, ESAT-PSI
Tinne.Tuytelaars@esat.kuleuven.be

Cordelia Schmid
INRIA, LEAR
schmid@inrialpes.fr

## Abstract

*Most recent class-level object recognition systems work with visual words,* i.e., *vector quantized local descriptors. In this paper we examine the feasibility of a data-independent approach to construct such a visual vocabulary, where the feature space is discretized using a regular lattice. Using hashing techniques, only non-empty bins are stored, and fine-grained grids become possible in spite of the high dimensionality of typical feature spaces. Based on this representation, we can explore the structure of the feature space, and obtain state-of-the-art pixelwise classification results. In the case of image classification, we introduce a class-specific feature selection step, which takes the spatial structure of SIFT-like descriptors into account. Results are reported on the Graz02 dataset.*

## 1. Introduction

Many current object recognition systems use as basic components vector-quantized local features, also referred to as 'textons' [16], 'object parts' [7], 'visual words' [26], or 'codebooks' [15]. Local features can be extracted either at interest points (*e.g.* [7, 15]) or sampled densely (*e.g.* [6, 13, 19, 20, 24]). In the context of category recognition, dense sampling has been shown to be advantageous, since interest points do not capture the entire appearance of a class. In this paper we use dense features and present a new approach for creating visual vocabularies.

Vector quantizing dense descriptors is especially difficult due to the number of descriptors and their distribution in space – there do not exist separate clusters. Existing solutions which improve over standard k-means include the use of mean-shift based clustering [9], hierarchical k-means [21], agglomerative clustering [14], and randomized trees [20]. They all present a compromise between accuracy and speed. In this paper we introduce an accurate and fast solution which differs significantly from existing solutions, as it does not rely on clustering techniques. Furthermore, our approach allows to explore the feature space and neighbourhood relations between visual words.

The basic idea behind our approach is to directly discretize the feature space using a regular lattice, *i.e.*, in a fully data-independent way. Each dimension is divided into $k$ partitions, resulting in $N^k$ bins for a N-dimensional feature space. Several authors have claimed that this approach is unfeasible for high values of $N$, since the number of bins grows exponentially with $N$. Indeed, in a 128-dimensional space, even when only 2 subdivisions for each dimension are used, this results in more than $10^{38}$ bins. With 4 subdivisions for each dimension, used in our experiments, this number increases even further to $10^{77}$ bins. However, most of these bins are empty. This is a typical phenomenon for high-dimensional spaces, re-enforced by the fact that all our features lie on the unit hypersphere (due to normalization). Moreover, some feature dimensions are highly correlated, which makes many combinations very unlikely in real images. As a result, only a small fraction of the bins need to be actually stored and can easily be accessed with a hashtable.

In short, our method consists of the following steps. First, dense descriptors are extracted from an image and collected in a sparse histogram, storing only the non-empty bins using hashing techniques (see section 2). This representation allows us to explore the feature space and to obtain a better insight in how features are distributed over this space, both in general as well as for images of a specific object class (see section 3). Additionally, we construct a fixed-size class-specific visual vocabulary by selecting the most discriminative bins and their neighbours. We experiment with different neighbourhood definitions, reflecting the spatial structure of the feature space (see section 2.5). We show experimental results in the context of localization (pixelwise classification) as well as image classification, both in a weakly supervised setting (section 4). Section 5 concludes the paper.

**Previous work.** Laptev et al. [12], Levi and Weiss [17], and Shoton et al. [25] have also proposed to start from densely sampled image patches. They then select discriminative descriptors using boosting. However, they heavily rely on the provided segmentations for the training data,

[*]Work performed during stay at INRIA.

whereas our method works in a weakly supervised setting. We do not rely on bounding boxes or segmentations during training but only use class labels. Moreover, our method is more generic, in that it provides a probability distribution as well as a visual vocabulary, which can then be used as input for any local features based object recognition system.

Lookup tables have been used before for object recognition. The best known example is probably the work on geometric hashing by Lamdan and Wolfson [11]. However, there the lookup table is not the actual image representation. The same holds for locality sensitive hashing, which has mostly been applied in the context of database retrieval [3].

Probably most similar to ours is the work of Jurie *et al.* [20] on building fast and discriminative visual codebooks using randomized clustering forests. They do not perform a traditional clustering of the feature space, but divide it in bins based on a set of randomized trees. The trees are selected so as to ensure high discriminativity. This allows for fast recognition, while maintaining state-of-the-art performance. We compare to their approach with respect to recognition accuracy (section 4) as well as computational complexity (section 2.6).

Several authors [13, 24, 27] have proposed to integrate the clustering with the subsequent classification problem, as we do in our feature selection (section 2.4). This allows to improve the classification results, but the resulting vocabulary is tuned towards the classification method used, and may be less suited for other methods.

In a sense, our work is also akin to the work of Konishi and Yuille [10], who investigate the use of colour and texture filter output statistics to perform a pixelwise classification of images into road, building, edge, vegetation, air, and other. Yet, they report problems with less homogeneous classes such as buildings. In contrast, our high-dimensional gradient orientation based descriptors are well suited especially for these structured classes.

## 2. Our Approach

### 2.1. Initial image description

We start by densely sampling the image – and when we say 'densely', we really mean 'densely': we use highly overlapping patches, with a spatial sampling rate starting at 1 pixel for the lowest scale, and a scalar sampling rate of 1.2, starting from $16 \times 16$ patches up to patches of size $120 \times 120$. This results in $1.433.254$ patches for an image of size $640 \times 480$. The reason for this very dense sampling is that we want to extract sufficient statistics from the training images to obtain accurate probability distributions and to limit the effect of discretization errors. For the time being, the same sampling scheme is also used during recognition. This can probably be relaxed, resulting in faster recognition performance, with minimal loss in performance.

Each of these patches is then described by a robust SIFT-like descriptor [18], computing distributions over gradient orientations for different subpatches. We experimented with $N_s \times N_s = 2 \times 2$ and $4 \times 4$ subpatches, with $N_o = 8$ different discretization levels for the orientations, resulting in feature vectors of respectively $N = N_s \times N_s \times N_o = 32$ and $128$ dimensions. To keep the computation time reasonable, we do not smooth the images (in contrast to SIFT). Instead, we compute the gradients only once at a single scale. Then, we use integral images to efficiently add up gradient magnitudes corresponding to a given orientation over a subpatch of our descriptor, in the spirit of [2]. The resulting feature vectors are subsequently normalized, except for homogeneous patches (*i.e.* those patches with all elements in the feature vector below a predetermined threshold), which are all set to zero.

### 2.2. Discretizing high-dimensional feature spaces

Each dimension of our feature space is then discretized in four different levels, such that it can be described by two bits. This results in no less than $4^N$ bins (*i.e.* on the order of $10^{19}$ or $10^{77}$ respectively for $2 \times 2$ and $4 \times 4$ subpatches) – a huge visual vocabulary indeed.

A square grid may not be the optimal lattice [1] for our feature space, with SIFT-like descriptors, which are typically compared using Euclidean distance (but see also section 2.5). Nevertheless, we prefer this simple lattice structure as it is more intuitive and allows for easy interpretation and exploration of the feature space (see section 3).

### 2.3. Practical implementation: Hashing

We exploit the fact that most of the bins after a direct discretization of the feature space are empty. With 2 bits for each dimension, a bin can be specified using an index of $2N$ bits. Rather than storing all $N^4$ possible bins in memory, we use a hash table and store only the non-empty bins. Constant time table lookup, *i.e.*, independent of the size of the visual vocabulary, can then be guaranteed.

To reduce problems with hash collisions, we use *double hashing* [8]. The original index $x$ of $2N$ bits is hashed using a first hash function $h_1(x)$. This value is used as a starting point in our hashtable. We, then, repeatedly advance steps of $h_2(x)$ to another address until the desired value is located or an empty location is reached. By using varied step sizes, determined by the second hash function $h_2(x)$, double hashing avoids the problem of *clustering*, *i.e.*, a high probability of repeated collisions. Constant-time table lookup is then guaranteed and only depends on the load factor $\alpha$, defined as the ratio between the number of elements to be stored and the size of the hash table. To ensure that search requires less than $t$ comparisons on average, it suffices to set $\alpha < (1 - 1/t)$. Double hashing has been shown to be asymptotically equivalent to the ideal uniform hashing [8]. If, for example, $\alpha = 0.1$, the chance of having a collision is one in ten but the chance that more than 10 addresses need to be checked is only $\alpha^{10} = 10^{-10}$.

We use a first hash table to store the number of occurrences of a particular bin, *i.e.*, the feature histogram or bag-of-words representation of an image or a set of images. The same data structure is also used to store for each bin the probability for a given class, the discriminativity (see below), or the visual word ID.

## 2.4. Feature selection

Our very high-dimensional bag-of-features representation can be used to explore the feature space (see section 3), or to perform weakly supervised pixel-wise classification (see section 4). However, the histogram is simply too large to be used as input for standard classifiers such as support vector machines. We have, therefore, implemented a mechanism to reduce the visual vocabulary to a predefined size. Note that we do not merge bins as in [27], but instead *select* the most *discriminative* bins (and their immediate neighbours) and ignore all the others. This is in contrast to existing visual vocabularies, where *all* features are assigned to their closest cluster center irrespective of the actual distance. This is reasonable in our case, as due to the dense sampling most of the features are very basic and not discriminative at all, see section 3. It is, therefore, better to ignore them right away. Discriminativity is measured by comparing the relative frequencies of a bin $x$ for a set of images of an object class and a set of background images:

$$r = \frac{p(x|class)}{p(x|background)}$$

It has been argued that feature selection prior to learning a support vector machine does not improve performance [4]. However, this is based on the assumption that the SVM can be trained on the entire dataset, which does not hold in our case, *i.e.*, for extremely high-dimensional vectors.

## 2.5. Neighbourhood definition

During the feature selection process, we exploit the fact that thanks to our regular lattice discretization, it is easy to retrieve neighbouring bins in feature space. However, not all $2^N$ neighbours are equally similar, due to the spatial structure of SIFT-like descriptors. Indeed, some dimensions are 'closer', *e.g.*, dimensions corresponding to similar orientations in adjacent subpatches, or adjacent orientations within the same subpatch. This affects the chance that a feature lie in the other bin due to discretization errors. We define four different types of 'neighbours' (see also figure 1):

- *Euclidean neighbours (E)*: identical descriptors, except for increasing or decreasing one element with one bit (one discretization level),

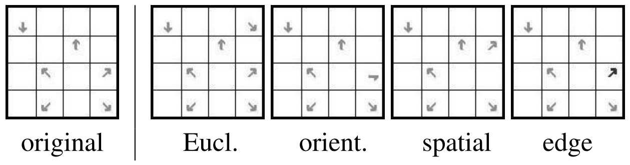

Figure 1. Examples of the different types of SIFT neighbours (darker arrows indicate stronger edges).

- *Orientation neighbours (O)*: identical descriptors, except for moving one bit from one element to another from the same subpatch and with adjacent orientation,
- *Spatial neighbours (S)*: identical descriptors, except for moving one bit from one element to another with the same orientation and from an adjacent subpatch,
- *Edge strength neighbours (D)*: identical descriptors, except for changing one non-zero element by one bit.

Note that edge strength neighbours are a subset of the Euclidean neighbours. Intuitively, spatial and orientation neighbours have a higher similarity with the original feature vector than some of the Euclidean neighbours. Nevertheless, their Euclidean distance is larger.

The number of nearest neighbours depends on the neighbourhood type, and also on the actual feature vector (since we have to stay within the four discretization levels). On average, for the $4 \times 4$ patches there are 131 Euclidean neighbours, 11 orientation neighbours, 16 spatial neighbours, and 10 edge neighbours. For the $2 \times 2$ patches, these numbers are 36 and three times 11 respectively.

A reduced visual vocabulary can then be constructed as follows. We select the most discriminative bin, assign it an ID, assign the same ID to its immediate neighbours, and remove these bins from the original hashtable. We iterate this process until a vocabulary of the desired size is obtained. By varying the neighbourhood type, we effectively vary the *specificity* of the visual words.

## 2.6. Complexity analysis

Building the full visual vocabulary (without feature selection) only involves hashing. Learning the distribution of features over the lattice takes $O(N_{train}D)$, with $N_{train}$ the number of patches in the training images and $D$ the dimensionality of the feature space. The dependency on $D$ is due to the computation of the descriptor, the discretization step, as well as the computation of the hash functions. Testing takes again $O(D)$ per feature or $O(N_{test}D)$ for all $N_{test}$ patches in the test image(s).

To build the fixed-size, class-specific visual vocabulary, we first build the full visual vocabulary, as above. Next, we compute the discriminative power for each bin. This step is linear in the size of our hashtable $H$, which has a predetermined fixed size. In our experiments, we typically set $H = 5 \times 10^7$ to guarantee a low load factor and a limited

Table 1. Average within-cluster standard deviation.

| | Nb. of clusters | Avg. standard deviation |
|---|---|---|
| KMeans | 100 | 0.612 |
| KMeans | 1.000 | 0.549 |
| KMeans | 10.000 | 0.503 |
| Ours | | 0.397 |

number of collisions. We then select the top $B$ most discriminative bins, again with complexity $O(H)$. The value for $B$ is chosen such that it is slightly larger than the expected number of bins in our final vocabulary. The top $B$ bins are sorted ($O(B\ log(B))$) and the top $k$ most discriminative bins are iteratively selected and added to the final visual vocabulary together with their neighbours. With on average $m$ neighbours, this takes $O(kmD)$. Since $B$ is only a fraction of $N_{train}$ and also $H$ is typically significantly smaller than $N_{train}$, the whole training procedure takes on the order of $O(N_{train}D)$. Testing involves only feature extraction, $O(N_{test})$, and hashing, $O(N_{test}D)$.

This is comparable to the complexity reported for randomized trees in [20], where training takes $O(\sqrt{D}N_{train}log(k))$ and testing $O(N_{test}log(k))$, with $k$ the number of clusters/leaf nodes before pruning.

## 3. Exploring the Feature Space

We now investigate the distribution of the features, both in general as well as for images of a specific object class. This allows us to analyze the structure of the feature space, and answer some intriguing questions. For convenience, we only report results on the 128-dimensional feature space here. Statistics are derived based on the Graz02 dataset [22].

***Question 1: How empty is the feature space ?*** For a specific object class, we obtain on the order of $10^6$ non-empty bins. For the more varied background images, this increases up to $10^7$. Compared to the total number of bins spanning the entire unit hypercube ($10^{77}$), this confirms our expectations that indeed most of the space is empty.

***Question 2: How 'compact' are these bins, compared to results obtained with k-means ?*** To measure the *compactness* of a bin, we compute the mean and standard deviation of the features it contains. With our direct discretization approach, we obtain an average standard deviation of $0.397$. This value is significantly lower than the average standard deviation obtained with K-Means, even when a relatively large number of clusters is used, as shown in table 1. This indicates that our visual words are more *specific*.

***Question 3: How are the features distributed over the bins ?*** As shown in fig. 2, some bins contain a very large number of features. However, the number of features per bin decreases fast. The most frequently visited bins correspond to very simple image structures, such as homogeneous patches and simple edge or line structures. More complex patterns typically occur less frequently (see fig. 2).

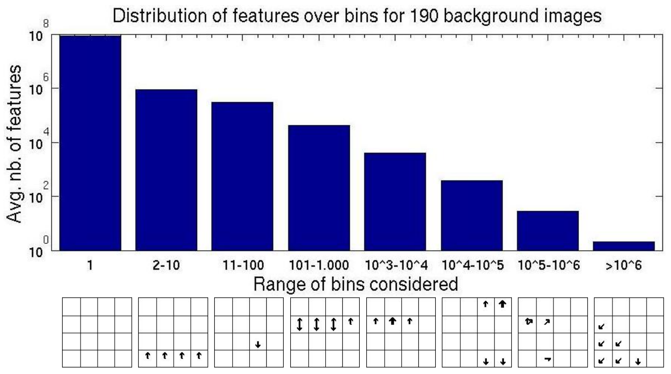

Figure 2. Top: Average number of features per bin for the 190 odd-numbered background images, with bins sorted by the number of features they contain (note the logarithmic scales). Bottom: For each range of bins, we visualize the central bin (*i.e.* bin nb. 1, 5, 50, 500, ...) to give an indication of the complexity of the image structures they represent (thicker arrows indicate stronger edges).

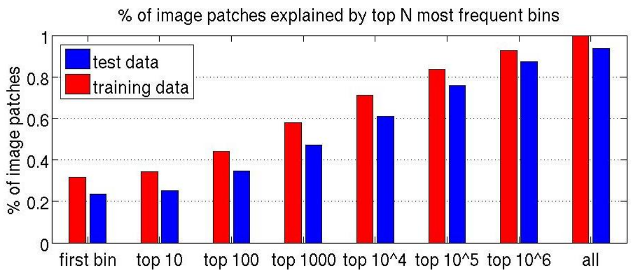

Figure 3. Number of image patches 'explained' by top N most frequent bins.

***Question 4: Is it sufficient to focus on the top N most frequent bins only ?*** This depends on the application. Even though some bins are very frequent, they still only represent a limited percentage of the data. To highlight this phenomenon, we display the information of fig. 2 differently in fig. 3. There seems to be a logarithmic relation between the percentage of image patches explained and the number of bins needed for this. For this experiment, we again use the 190 odd-numbered background images. We sort the bins based on frequency, and then check which percentage of the image patches falls within the top $N$ most frequent bins, for varying values of $N$. The first bin already explains about $30\%$ of the data. This does not come as a surprise, since it corresponds to homogeneous or near-homogeneous patches which indeed abound in many images. With the top 10.000 bins, over $75\%$ of the image patches is accounted for. However, many more bins are needed to cover the remaining $25\%$. If we repeat the same experiment with image patches from the test set (the 190 even-numbered background images), we get approximately the same result (see fig. 3), except that we never reach $100\%$: $6\%$ of the patches are in bins that were never activated in the training images.

***Question 5: How similar are distributions learned over different image sets ?*** To evaluate the similarity between

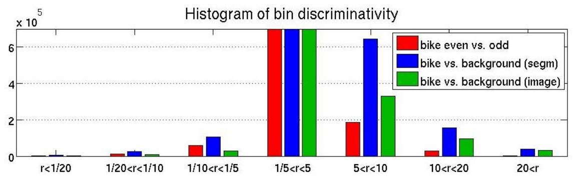

Figure 4. Histogram over bins based on discriminativity $r$.

two distributions $d_1$ and $d_2$, we compute for each bin its *discriminativity* $r$, *i.e.*, how more likely is it to find a feature belonging to this bin in images of $d_1$ than in images of $d_2$. The number of highly discriminative bins then gives an indication of how (dis)similar the two distributions are. First, we compare the distributions computed over the odd-numbered and even-numbered bike segmentations (see fig. 4). Only a limited number of bins have a discriminativity larger than 10 or smaller than $1/10$. So within a particular category, distributions learnt over different images sets are quite similar. When comparing the distribution of the bike segmentations with the distribution of background images, on the other hand, many more discriminative bins appear. So the learnt distributions have a class-specific component. These are the important bins to discriminate between bikes and background. Finally, we repeat the same experiment but now using the entire bike images instead of the segmentations. The most discriminative bins ($r > 20$) are relatively stable. This suggests a weakly supervised setting may suffice in some settings.

***Question 6: What are the most discriminative bins ?***

Fig. 5 visualizes some of the most discriminative bins for the object classes bike, car, and person (extracted in a weakly supervised setting). The most discriminative patches for bikes contain diagonal structures. Somewhat surprisingly, our method does not find the wheels among the most distinctive patterns for cars. This is in contrast to what has been observed in many interest point based methods. Instead, combinations of horizontal and diagonal edges, typical for car windows, are most discriminative. This may be due to the fact that the typical bins representing wheels are not very frequent for cars (with at best 2 occurrences per car). For persons, non-perfect vertical structures (*e.g.* on people's legs) and rounded structures score well. The most discriminative bins have a relatively high complexity.

***Question 7: Are these results data-dependent ?***

Finally, it is important to check whether these results are generic or not. To this end, we repeat the same analysis as above on the MSRC dataset [25]. This time, we do use the segmentations, as each image contains multiple object classes. Due to space constraints, we cannot show all the graphs, but the overall trends and conclusions are the same, except maybe for question 5. We observe that the 'overlap' between distributions of different categories is much larger. Some of the classes, like grass and road, are rather textures than true object categories. If the texture is too finegrained,

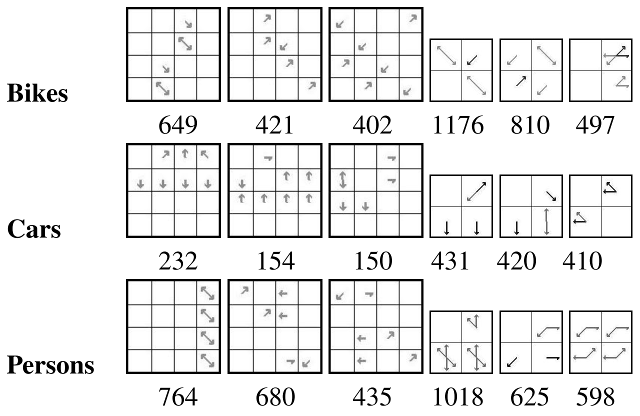

Figure 5. Visualization of the most discriminative bins, computed over the odd-numbered images of the Graz02 dataset. The discriminativity is mentioned underneath each visualization.

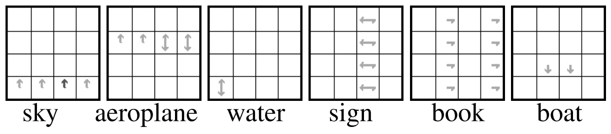

Figure 6. Visualization of the most discriminative bins for some of the MSRC categories.

it cannot be captured by our descriptors. Moreover, often color is the only cue to discriminate between these classes, and this is currently not included in our descriptor. For the more structured classes, like book, building, or aeroplane, on the other hand, the results are more similar to the results on the Graz02 dataset.

In figure 6 we show the most discriminative bins for some of the MSRC categories. The most discriminative bin for sky seems to capture horizons. Other highly discriminative bins for sky (not shown here) are triggered on cloud fragments. For aeroplanes, horizontal line structures seem important. The two opposite arrows in the same subpatch in the most discriminative bin for water correspond to small ripples on the water. For signs, vertical line structures as can be found in many letters are a good cue. The book images mainly contain book shelves, with many parallel vertical lines, and these can be detected well with the fifth bin of figure 6. For boats, finally, short horizontal structures are important. These are found a lot both on the boat as in the surrounding water, which is sometimes included in the segmentation for boat.

## 4. Experimental results

From the above discussion, it is clear that by direct discretization an accurate, albeit large visual vocabulary can be obtained. Moreover, we have observed that a large number of features reside in a few bins, corresponding to very simple patterns. They are not really informative nor discriminative. As a result, they do not contribute much to image classification. This justifies our approach of feature selection based on discriminativity.

On the other hand, using hashing techniques, lookup times do not depend on the size of the visual vocabulary. Hence we only apply such feature selection when a smaller vocabulary is really needed, *e.g.* to train a support vector machine classifier. For other applications, such as the pixelwise classification, the full visual vocabulary can be used.

For our experiments, we use the Graz02 dataset [22]. For the first 300 images of each class, segmentations are provided. We use these only for the evaluation of the pixelwise classification, and never for training. Unless otherwise mentioned, we use odd images for training and even images for testing. Default parameters are $4 \times 4$ patches and, in case of feature selection, $10.000$ visual words including edge strength, orientation and spatial neighbours.

**Use of resources.** For an image of size $640 \times 480$ and using a standard PC, it takes $0.26ms$ to compute the integral images. Computing the descriptors for 1000 features then takes an additional $12ms$, discretizing them $1.4ms$, and the final hashing less than $0.5ms$. On the downside, the hashtables take a lot of memory, especially for the full visual vocabulary: we use a hashtable of size $5 \times 10^7$, which results in a file of $5 \times 10^7 \times (4 \times 4 \times 8 \times 2 + 32)/8 = 1.8Gbyte$. After feature selection, this can be reduced to about $20Mbyte$ for a visual vocabulary of size 1000. This is a result of the hashing approach, which tries to find a tradeoff between memory and speed, by avoiding collisions. A linked-list hash table would probably take less space, but can in a worst case scenario result in a time complexity linear in the number of features stored in the hashtable.

## 4.1. Pixelwise classification

Based on the distributions learnt during training, each patch can be classified as being more or less likely to belong to a specific object class, by retrieving the discriminativity (or likelihood ratios) of the bin it falls in from the corresponding hash tables. To turn these patch-wise classifications into a pixel-wise classification, we make the simplifying assumption that the likelihoods for all patches are independent, even if they largely overlap. The likelihood for a specific pixel is then given by the product of the likelihoods of all patches containing that pixel. This results in likelihood maps as shown in figure 7. Note that these are obtained in a weakly supervised manner, i.e., without using the segmentation masks or the bounding boxes. Nevertheless, the system has learnt which parts of the images are relevant for a specific class, resulting in relatively accurate segmentations, in spite of a wide range of viewpoint and intra-class variations. This is remarkable, since only local image information is exploited, without imposing any spatial smoothness or consistency.

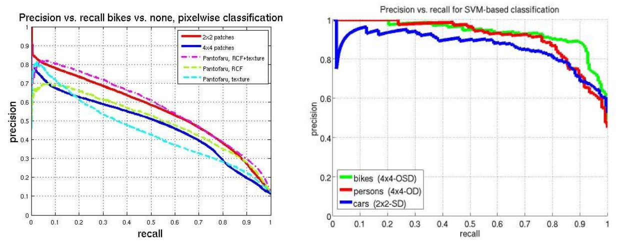

Figure 8. Precision-recall curves for pixelwise classification (left) and image classification (right)

Table 2. Comparison of classification rate with state-of-the-art

| Method | Bikes | Cars | Persons |
|---|---|---|---|
| Opelt et al. [22] | 76.5 | 70.7 | 81.0 |
| Jurie et al. [20] | 84.4 | 79.9 | / |
| Ours | **89.5** | **80.2** | **85.2** |

Even though the background class was conceived to be unbiased, it contains some context information. For instance, road markings have a high probability under the car model, and so do bicycle racks for the bicycle model, as can be seen in some of the examples of figure 7.

To quantitatively evaluate the performance of the pixelwise classification, we show the precision-recall curve for the bikes in figure 8. Somewhat surprisingly, $2 \times 2$ patches seem to outperform their $4 \times 4$ counterparts in this experiment. We also show the result obtained by Pantofaru *et al.* [23]. Even though we only use local measurements, and not information from a larger context and/or image segmentation as they do, similar performance is obtained. Using image segment classification as an intermediary step between patch-wise classification and pixel-wise classification is likely to give a further increase in the classification accuracy. Finally, the quality of these likelihood maps could be improved further by taking into account spatial consistency, *e.g.* using Markov Random Fields.

## 4.2. Image classification

Based on the reduced visual vocabulary construction explained in section 2.4, we can build a more compact bag-of-words representation of the image, and use that to train a SVM-classifier. In our experiments, we used SVMs with a linear kernel. This results in EER rates of $89.5\%$ for bikes, $80.2\%$ for cars, and $85.2\%$ for persons (each time for the optimal parameter settings, see below). To our knowledge, this outperforms previously reported classification results on this dataset, as can be seen from table 2. Precision versus recall is shown in fig. 8.

**Effect of parameter settings** Table 3 summarizes the effect of some of the parameter settings. For SVM-based classification, $4 \times 4$ patches slightly outperform $2 \times 2$ patches

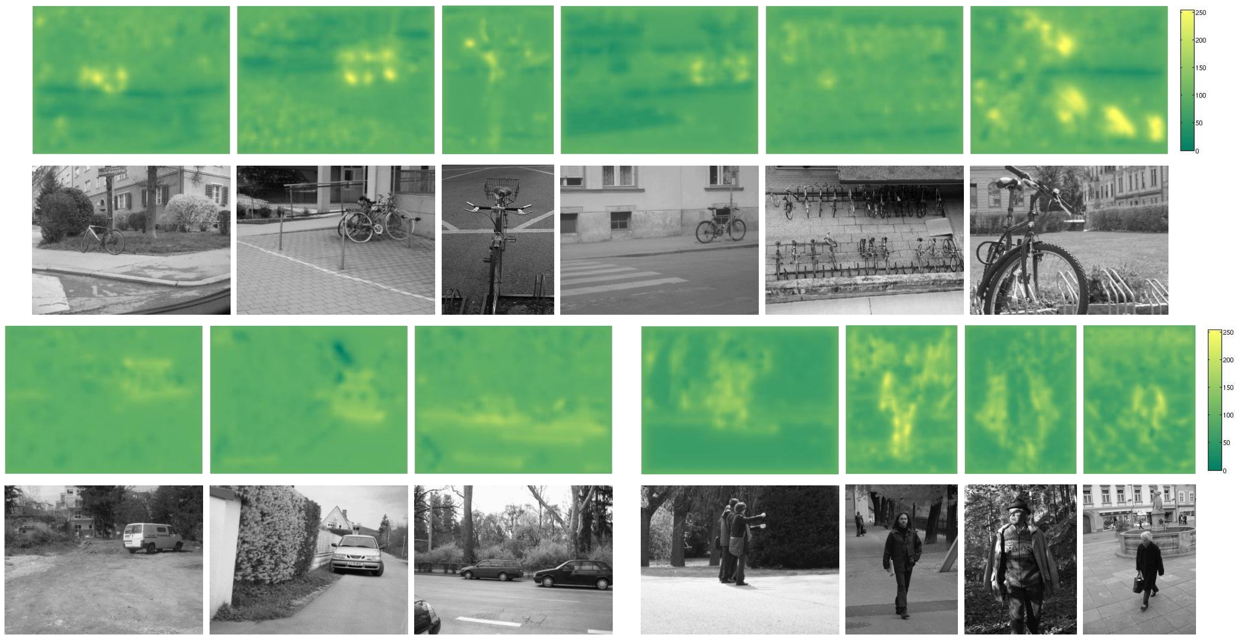

Figure 7. Pixelwise classification for a few example images (based on $4 \times 4$ patches). (Top): Bikes, (Bottom Left): Cars, (Bottom Right): Persons. Dark green means a negative log likelihood (unlikely to be the object of interest); Light green and yellow indicate a positive log likelihood. Zero log likelihood corresponds to value 100 on the colorbar (best viewed in color).

Table 3. Effect of different parameter settings. ('E' refers to Euclidean neighbours, 'D' to edge strength neighbours, 'O' to orientation neighbours, 'S' to spatial neighbours - see also figure 1)

| neighb. type | **Bikes** $4 \times 4$ | **Cars** $4 \times 4$ | **Pers.** $4 \times 4$ | **Bikes** $2 \times 2$ | **Cars** $2 \times 2$ | **Pers.** $2 \times 2$ |
|---|---|---|---|---|---|---|
| / | 86.0 | 71.8 | 81.5 | 79.6 | 68.3 | 76.2 |
| E | 85.6 | 77.0 | 84.1 | 87.1 | 78.5 | 82.0 |
| D | 88.1 | 77.7 | **85.2** | 87.4 | 78.5 | 82.9 |
| OD | 87.4 | 77.5 | **85.2** | 86.0 | 71.5 | 84.6 |
| SD | 88.4 | 77.5 | 84.1 | 86.8 | **80.2** | 84.3 |
| OSD | **89.5** | 76.1 | 84.6 | 87.1 | 79.7 | 84.6 |

Table 4. Effect of visual vocabulary size and neighbour type for the Graz02 bikes - same naming convention as in table 3.

| | 50 | 100 | 500 | 1000 | 5000 | 10.000 |
|---|---|---|---|---|---|---|
| / | 66.4 | 76.1 | 85.2 | 87.9 | 85.2 | 86.0 |
| **E** | 78.7 | 81.4 | 82.8 | 82.5 | **88.3** | 85.6 |
| **D** | 77.5 | 82.0 | 83.6 | 82.2 | 87.2 | **88.1** |
| **OD** | 72.0 | 82.2 | 84.1 | 81.8 | **89.2** | 87.4 |
| **SD** | 78.11 | 83.1 | 83.6 | 85.5 | **88.5** | 88.4 |
| **OSD** | 77.7 | 83.6 | 83.9 | 85.8 | **89.5** | 89.0 |

for bikes and persons, but not for cars. Adding neighbours helps to improve the results with a few percent. Orientation and/or spatial neighbours typically yield slightly better results than what is obtained with Euclidean neighbours, but the difference is not significant.

Table 4 shows the effect of changing the size of the visual vocabulary. Note that, contrary to traditional clustering approaches, in our case the average number of features for a visual word is determined solely by the neighbourhood type, and does not depend on the size of the vocabulary. Indeed, the smaller visual vocabularies are just subsets of the larger ones. As a result, one can expect the larger vocabularies to perform better: they contain more information, and the SVM classifier is able to ignore the redundant features. This is indeed what happens. Nevertheless, at a certain point the newly added features do not have much discriminative power, and the curves saturate or even degrade slightly. Remarkably, vocabularies of just 50 or 100 features already achieve state-of-the-art results. Moreover, whereas adding neighbours increased the classification accuracy for large size vocabularies, the opposite is true for smaller size vocabularies ($500 - 1000$). More specific visual words seem to be advantageous in that case. But when the size of the vocabulary gets really small, this strategy breaks down, and adding neighbours again works better.

**Other datasets.** We did some preliminary experiments on the MSRC and Pascal VOC 2006 datasets [25, 5], using the same parameter settings and setup as for Graz02. However, in contrast to Graz02, they do not have a clean background dataset and, as observed in section 3, the categories show larger overlap in their distributions. This affects the feature selection step and, as a result, also the performance of the final classifiers (which gave slightly above average results compared to all Pascal VOC participants). Exploiting the bounding boxes during training and/or more parameter tuning could probably improve these results.

## 5. Conclusion

To summarize, we have developed an efficient method for vector quantizing feature space which is independent of the training data. Our approach directly discretizes the feature space using a regular lattice. In spite of the high dimensionality the access is fast thanks to hashing, as the space is mainly empty. By using a regular lattice we stay close to the original SIFT-like representation. This allows for easy interpretation of visual words, for fast retrieval of neighbouring bins in feature space, and for exploring the structure of the feature space. Additionally, we experiment with different types of neighbours, that exploit the spatial structure of SIFT-like descriptors. State-of-the-art results are demonstrated both for pixelwise classification as well as image classification.

As future work, we plan to experiment with other descriptors, including different shapes of patches, as well as combinations of different descriptors. Another line of research involves bringing in spatial information and spatial consistency for improved pixelwise classification.

**Acknowledgements** The authors acknowledge support from European project CLASS, Network of Excellence PASCAL, the INRIA visiting professor program, and the Fund for Scientific Research Flanders.

## References

[1] E. Agrell and T. Eriksson. Optimization of lattices for quantization. *IEEE Transactions on Information Theory*, 44:1814–1828, 1998.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[3] T. Darrell, P. Indyk, and G. e. Shakhnarovich. Nearest neighbor methods in learning and vision: Theory and practice. *MIT Press*, 2006.

[4] G. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. In *ICCV*, 2005.

[5] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results.

[6] L. FeiFei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[7] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, volume 2, pages 264–271, 2003.

[8] L. Guibas and E. Szemeredi. The analysis of double hashing. *Journal of Computer and System Sciences*, 16:226–274, 1978.

[9] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.

[10] S. M. Konishi and A. Yuille. Statistical cues for domain specific image segmentation with performance analysis. In *Conf. on Computer Vision and Pattern Recognition*, 1999.

[11] Y. Lamdan and H. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Int. Conf. on Computer Vision*, pages 238–249, 1988.

[12] I. Laptev. Improvements of object detection using boosted histograms. In *British Machine Vision Conference*, volume III, pages 949–958, 2006.

[13] D. Larlus and F. Jurie. Latent mixture vocabularies for object categorization. In *British Machine Vision Conf.*, 2006.

[14] B. Leibe, K. Mikolajczyk, and B. Schiele. Efficient clustering and matching for object class recognition. In *British Machine Vision Conference*, 2006.

[15] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *British Machine Vision Conf.*, 2003.

[16] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using threedimensional textons. *Int. Journal of Computer Vision*, 43(1):29–44, 2001.

[17] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 53–60, 2004.

[18] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[19] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *Conf. on Computer Vision and Pattern Recognition*, pages 34–40, 2005.

[20] F. Moosmann, B. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. In *NIPS*, 2006.

[21] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.

[22] A. Opelt and A. Pinz. Object localization with boosting and weak supervision for generic object recognition. In *Scandinavian Conference on Image Analysis*, 2005.

[23] C. Pantofaru, G. Dorkó, C. Schmid, and M. Hebert. Combining regions and patches for object class localization. In *CVPR06 Beyond Patches Workshop*, 2006.

[24] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. In *European Conference on Computer Vision*, 2006.

[25] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, 2006.

[26] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, volume 2, pages 1470–1477, 2003.

[27] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *International Conference on Computer Vision*, 2006.