# Vehicle Categorization: Parts for Speed and Accuracy

Eric Nowak, Frédéric Jurie

# Vehicle Categorization: Parts for Speed and Accuracy

Eric Nowak[1,2] Frédéric Jurie[1]

[1] Laboratoire GRAVIR / UMR 5527 du CNRS - INRIA Rhone-Alpes - UJF - INPG
[2] Société Bertin - Technologies, Aix-en-Provence
{eric.nowak,frederic.jurie}@inrialples.fr

## Abstract

*In this paper we propose a framework for categorization of different types of vehicles. The difficulty comes from the high inter-class similarity and the high intra-class variability. We address this problem using a part-based recognition system. We particularly focus on the trade-off between the number of parts included in the vehicle models and the recognition rate, i.e the trade-off between fast computation and high accuracy. We propose a high-level data transformation algorithm and a feature selection scheme adapted to hierarchical SVM classifiers to improve the performance of part-based vehicle models.*

*We have tested the proposed framework on real data acquired by infrared surveillance cameras, and on visible images too. On the infrared dataset, with the same speedup factor of 100, our accuracy is 12% better than the standard one-versus-one SVM.*

## 1. Introduction

Identifying objects captured by a video-camera is a classical application for video-surveillance. Once an object is detected on an image, a classification algorithm analyzes its region of interest (ROI) to predict the associated object class. In this study we focus on this classification. We propose a framework for fast and accurate classification of different types of vehicles images in a natural cluttered background.

Recent state-of-the-art methods are either template or appearance based. In template based methods [8], recognition is based on distances (e.g. correlations) between an input image and the predefined templates. Appearance-based methods [1, 6] learn the variability of vehicle appearances from a set of training examples. Images are represented by a set of local (so-called part-based approaches) or global features. The advantage of part-based methods is their robustness to pose, illumination and shape variations, and partial occlusions. A part based object recognition system consists of three steps [1, 14, 6]. In figure 1 we illustrate the first step which selects the most useful parts. The example shows discriminative object parts and their detected locations on car images. The second step combines these parts into object representations, and the last builds a classifier for the recognition system. The selection of these parts is critical. The system has to choose those *few* parts that are discriminative enough to separate between classes.

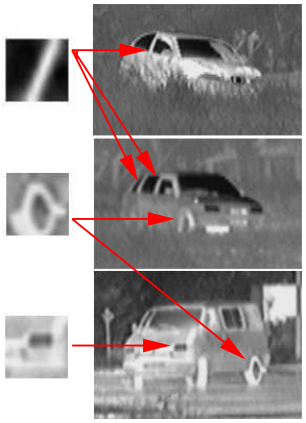

Figure 1: Example of parts. Which ones best discriminate between classes?

To identify an part by correlation is computationally expensive, therefore this paper proposes a new method to select these very few but useful parts. As a first contribution, our approach takes advantage of the tree structure of hierarchical classifiers by individually selecting parts for each classifier-node (see section 3.4). The second contribution is a new high-level data representation that increases the recognition accuracy (see section 3.2).

The organization of the paper is as follows. Section 2 describes part-based object classification methods of the state-of-the-art. Section 3 presents our method. Experiments are in section 4.

## 2. Part-based object classification

In this section, we describe state-of-the-art part-based object classification methods. We detail the phases of the process: visual codebook computation, part detection, image representation and multi-class classification.

## 2.1. Visual codebook computation

A part is a representative of local image descriptors that appear frequently in a given dataset. For example, on figure 1, the wheels on the second and third car image are represented by the second part. A visual codebook is the set of parts defined to describe the images of a dataset. It is generally obtained by a quantization algorithm. First, local image descriptors are computed at different locations and scales, and then they are grouped by their similarity. Each group is represented by a so-called part.

Leug and Malik [7] use a filter bank convolution to describe regularly sampled pixels, and k-means to compute the parts. Agarwal [1], Leibe [6] use interest point detectors to determine characteristic locations which are then described by the gray-level values of a fixed size neighborhood. They build their codebook by an agglomerative clustering method. For quantization, Willamowski [15] uses k-means and Jurie [5] uses a mean-shift based density estimation.

## 2.2. Part detection

[1, 6, 15] use interest point detectors, therefore consider the images at sparse locations. The detection process is fast, but on the other hand it performs poorly if the object is small, or if the informative regions are not detectable by the chosen interest point operator. [11, 14, 5] suggest to use a dense multi-scale representation, where the patches are detected on images by multi-scale correlation. This step is computationally expensive, therefore, real-time systems require to reduce the size of their codebooks (see 3.3).

## 2.3. Image representation

Agarwal [1] and Leibe [6] use geometric constraints to model the relation between parts. Agarwal takes into account pairwise relations between parts (distance, orientation). Leibe models the position of parts with respect to the center of the object they belong to. Alternatively, Willamowski's *bag of features* approach [15] ignores the geometric relations between parts and images are represented by the occurrences of codebook entries. This simple representation gives surprisingly good results in [5].

## 2.4. Multi-class classification

This section presents different state-of-the-art multi-class classifiers adapted to "bag of features" image representations.

Decision trees [2] are popular multi-class classifiers. Because their construction is based on joint probabilities, other classifiers are a better choice when the dimension of the feature space is high and the training examples are limited.

The most popular choice of generative classifier in case of high dimensional feature space is the Naive Bayes classifier, because it assumes that the features are independent, therefore it avoids to estimate the joint probability. The Naive Bayes classifier models the probability that a feature vector $V$ belongs to a category $C_k$ as $P(V|C_k) = \prod_j P(V_j|C_k)$. $P(V_j = 0|C_k)$ and $P(V_j = 1|C_k)$ are learned during the training phase. An unseen image described by a vector $V$ is affected to the category maximizing $P(V|C_k)$.

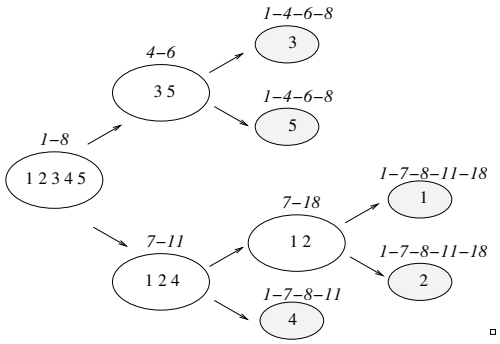


Figure 2: *HBC classifier principle.* The classes are iteratively separated. *Feature selection.* 2 features are selected per classifier (above the nodes); the features used to predict a class are indicated above the leaves

In recent articles the family of discriminative classifiers is dominated by SVMs. They were originally designed for binary classification problems [13, 12] and were extended to multi-class with "one versus one" (`1vs1`) and "one versus the rest" (`1vsR`) strategies. `1vs1` trains all pairwise combinations of objects. During prediction, the $C(C-1)/2$ binary classifiers are evaluated and the class that receives the majority of the votes is assigned to the test sample. `1vsR` trains a binary classifier for each class to separate its objects from the rest. The classifier with the highest confidence determines the label for an unseen test sample. Recently, Rajan [10] proposed a Hierarchical Binary Classifier (HBC) representing the multi-class problem by a binary tree. This is illustrated on figure 2. The root of the tree is the set of all classes. The nodes are iteratively partitioned in two sets until they contain only one class. At each node, a binary classifier separates the partitioned sets. To classify an unseen example, a path from the root to a leaf is followed in the tree, according to the predictions of the classifier-nodes. Rajan constructs the tree with an iterative k-means algorithm ($k = 2$).

# 3. The proposed method

In this section we present our framework for vehicle classification. We propose a new data transformation and an efficient integration of feature selection to HBC. First, we give an overview of the overall framework, and then detail each component.

### 3.1. Our framework

Our model is built from a training set of images as follows:

1. *Codebook computation.* Our image descriptors are dense, multi-scale, gray-level pixel intensities computed at local neighborhood. We use mean-shift quantization of [5] to obtain a codebook of size $n$.

2. *Representation of the training set.* We use a "bag of features" approach, and each image of the training set is represented by a vector of size $n$.

3. *Data transformation.* The vectors are transformed into a higher-level representation (see 3.2)

4. *Computation of a HBC classifier.* We use Rajan's iterative k-means to compute the tree adapted to our dataset [10].

5. *Feature selection adapted to the tree.* The most useful parts are selected for each classifier-node of the tree (see 3.4)

For prediction, we follow a path from the root of the HBC to a leaf as follows:

1. Start path at the root node.

2. Detect on the input image the selected parts for the the *current* node. Note: parts are selected during training, point 5.

3. Classify the image on the current node to determine which of the two child nodes is the next node of the path.

4. Loop over 2-3 until a leaf node is reached. The label of the leaf node is the predicted label of the image.

Below, we present our data transformation algorithm, we compare different feature selection strategies and explain how to use them with HBC. We explain why HBC classifiers perform better than other classifiers with small codebooks (later section 4.4 confirms it experimentally).

### 3.2. Data transformation

"Bag of features" approaches represent an image $I_i$ by a vector $V_i$, where $V_i(j)$ is the number of detections of the part $j$ on that image. This information can be transformed to improve classification results. Histogram estimation of bayesian classifier probabilities requires to quantize individually the values of each $V_i$. And because they use euclidean distances, linear SVMs also require a data transformation, as the following example shows. Let $d(x_i, x_j)^2 = \sum_{k=1..n} (x_i^k - x_j^k)^2$ be the distance between two support vectors. If the magnitude of the first dimension is higher than the other ones, the distance becomes $d(x_i, x_j)^2 \simeq (x_i^1 - x_j^1)^2$ and the information from other dimensions is lost.

Below we enumerate standard data transformation methods:

1. Original data (`raw`): vectors are not modified, classifiers use occurrences.

2. Linear transformation, by feature (`ranging`). Each feature dimension is linearly transformed into $[0, 1]$ in order to give the same importance to all the features.

3. Linear transformation, by image (`proba`). Each $V_i$ is independently normalized. After a L1 normalization, $V_i(j)$ is the probability of finding the part $j$ on the image $I_i$.

4. Simple binarization (`bin0`). If the feature $j$ appears at least once in the image, $V_i(j) = 1$ else $V_i(j) = 0$. We only consider that a feature appears, and not the number of times it appears.

We propose a higher-level transformation (`binAuto`), that automatically computes the optimal binarization threshold for each feature. This threshold is chosen among a list of candidates by a mutual information maximization process. This process measures how useful each binarization threshold is to the classification, and picks the most informative threshold. This definition is motivated by the following consideration. A feature appearing in all images of all categories is not discriminative. But if it appears often in many images of a category, and rarely in many images of other categories, "a minimum number of detections" is a discriminative information. The standard binarization (`bin0`) removes this information. The original data (`raw`) contains this information, but only implicitly. On the contrary, the automatically binarized data (`binAuto`) explicitly contains this discriminative information.

### 3.3. Feature selection

In this section, we present different feature selection strategies. In the next section we will explain how to integrate them efficiently to HBC classifiers.

Given a codebook of $n$ parts, feature selection tools [4] select the $n' < n$ most useful ones for classification. The optimal sub-codebook requires a NP-compete search, i.e. the examination of all codebook subsets of size $n'$. In practice this is not possible, and a sub-optimal search is used.

A wrapper method constructs the sub-codebook iteratively. The subset is initially empty. Iteratively, the part that gives the largest increase of performance is added to the subset. In section 4.3, we show that this method gives the best performance. The weak point of a wrapper method is its quadratic complexity ($O(nn')$).

A more greedy approach assumes that the features are independent. A utility measure is computed for each part,

and the optimal subset is made of the $n'$ top ranked features. The complexity of this algorithm is linear ($O(n)$). Many utility functions were proposed in text classification applications [9], here we detail the most commonly used measures:

1. Random selection (`rand`), to avoid local optimum problems and redundancy
2. Frequency (`freq`), to rely more often on available information.
3. Mutual information[1] (`MI`), to measure how useful a part is to the classification problem. Let $C$ be an object category, $V_j$ indicate if the part $j$ is detected on an object or not. The mutual information between $C$ and $V_j$ is defined as

$$I(V_j, C) = \sum_{v_j=0,1} \sum_{c=1..C} p(c) Freq_{j,c} \log(Discr_{j,c})$$

$$Freq_{j,c} = p(v_j|c) \ , \ Discr_{j,c} = \frac{p(v_j|c)}{\sum_c p(v_j|c)p(c)}$$

   MI combines frequency statistics of the part ($Freq$) and its discriminativeness ($Discr$).
4. Odds ratio (`OR`) selects the most discriminative features. However, as it ignores frequency, it does not allow a dramatic reduction of the codebook size. `OR` is defined as $\frac{p(v_i=1|c=1)}{p(v_i=0|c=1)} \frac{p(v_i=0|c=2)}{p(v_i=1|c=2)}$.
5. Linear SVM hyperplane normal coefficient (`omega`) focuses on discriminative parts due to the scalar product of the prediction function: $svm(V) = \omega V + b$, $V = [V_1 ... V_n]$. Thus, if all feature dimensions have the same range, more important features have higher associated hyperplane normal coefficient

### 3.4. Feature selection for HBC

Naive Bayes, `1vsR` and `1vs1` classifiers respectively require the evaluation of $C$, $C$ and $C(C-1)/2$ classifiers. If we run one of the previously described feature selection tools, we obtain a set of $n'$ features. These features are useful for the multi-class classifier, but not for *each* binary classifier. Thus, the selected features are useless during a major part of the multi-class classification. As HBC are evaluated sequentially, from the root to a leaf, they don't have this drawback. This is illustrated on figure 2. If the root node classifier predicts that the input image belongs to classes 1, 2 or 4, it is useless to detected the parts 4 and 6, as they are only useful to separates classes 3 and 5. Instead, the parts 7 and 11 are be detected to discriminate between the classes 1, 2 and 4. This process is computationally efficient because 7 different parts are used for classification, but on average an example is classified with $(4+4+5+5+4)/5 = 4.4$ parts.

For this reason, we propose the following feature selection method for HBC: use any of the methods presented in 3.3 to select $m$ features for each classifier-node.

## 4. Experiments

In the previous section, we have presented two methods to improve "bag of features" approaches: feature selection integrated to hierarchical classification and automatic data binarization. In the following experiments we will show the effects of each of these methods on an infrared video-surveillance problem. In order to prove that the method is not limited to the infrared sub-band, we will briefly show results on a visible light dataset.

### 4.1. Context

In this section we describe the datasets used to evaluate the performance of the recognition algorithms.

The first dataset is built from infrared images. An image contains one out of four models of vehicles (1 van, 3 cars) in a natural cluttered background. The vehicles are seen at different scales (100 to 600 meters), orientations, illumination conditions and occlusion levels. The dataset is made of approximate regions of interest (ROI) of the vehicles, as if the vehicles were detected by a realistic tracker (the red rectangles in figure 5). An approximate ROI is obtained by a random transformation of the exact ROI: its size is multiplied by $m$ in $[0.8, 2]$, then its position is shifted by $(s_w.w, s_h.h)$ pixels, with $s_w$ and $s_h$ in $[-0.4, 0.4]$, $w, h$ the width, height of the exact ROI. The dataset contains 250 images of each vehicle and 1500 images of background. The recognition algorithm has to discriminate between 4+1 classes.

The second dataset contains visible light images of 8 classes: horse, face, motorbike, car, bike, book, phone (50 images each) and background (400 images).

The prediction accuracy is evaluated by ten-fold cross-validation. The accuracy is the mean of all classes true positive rate.

### 4.2. Codebook computation

The codebook is computed with the algorithm mentioned in section 3.1. The NCC threshold is set to 0.8, which corresponds to an angle of 40 degrees between the two patches. The scale pyramid has 9 levels, and the scale ratio is 0.91. The mean-shift algorithm iteratively produces 4000 clusters — or parts — , the mean shift ball radius is set to 0.8. Figure 3 displays codebook patches frequently and rarely detected in the IR dataset.

### 4.3. Feature selection

In this section, we study the effects of feature selection strategies to decide which one to use in the HBC nodes. We

[1] called mutual information in [3] and information gain in [16]

Figure 3: Above (below) the 80 most (less) frequent words of the IR database

use `raw` data and a `1vs1` linear SVM classifier to compare the different methods. Figure 4 shows the mean recognition accuracy of each class w.r.t. the percentage of considered features.

We observe that the codebook contains useless (or redundant) words, because the `MI` feature selection method reaches the maximal performance with 40% of the vocabulary with the IR dataset, and 10% with the other dataset. The wrapper method is the most efficient feature selection method, because it explicitly optimizes the performance. Unlike the other ranking measures, the curve grows very fast. `OR` and `freq` give worse performance than `rand`. In the visible database, `omega` also gives worse performance than `rand`. This is not the case of `MI`, which is always better than `rand`: on the visible database, +20% with 1% of the features, +25% with 4% of the features, on the IR database +15% with 4% of the features.

We can conclude that the vocabulary is very redundant and it is possible to reduce its size and keep good classification performance in the same time. The wrapper method is the most efficient of the ones we studied; `MI` is the most efficient linear method. In the rest of the experiments, we only consider `MI` selection because of its lower computational complexity.

### 4.4. HBC and feature selection

In this section, we compare the effects of different multi-class strategies on the feature selection process. We want to measure the difference between Hierarchical Binary Classifiers and other classifiers. We compute two HBC trees: `tree1` is obtained by Rajan's algorithm [10], `tree2` is randomly built. `1vs1` and `1vsR` are the classical SVM multi-class classifiers, and `NaiveBayes` is the maximum likelihood Naive Bayes classifier. The classification results are reported on figure 6.

We first notice that `1vs1` and `1vsR` have a better accuracy than `tree` and `NaiveBayes` when we use 100% of the features (4000). However, because we are interested in computational efficiency, the most interesting part of the graphs is the one with small number of features. Then, `tree1` has a better accuracy than `1vs1` and `1vsR`. For example, in the IR database, `tree` is 10% better than `1vs1`

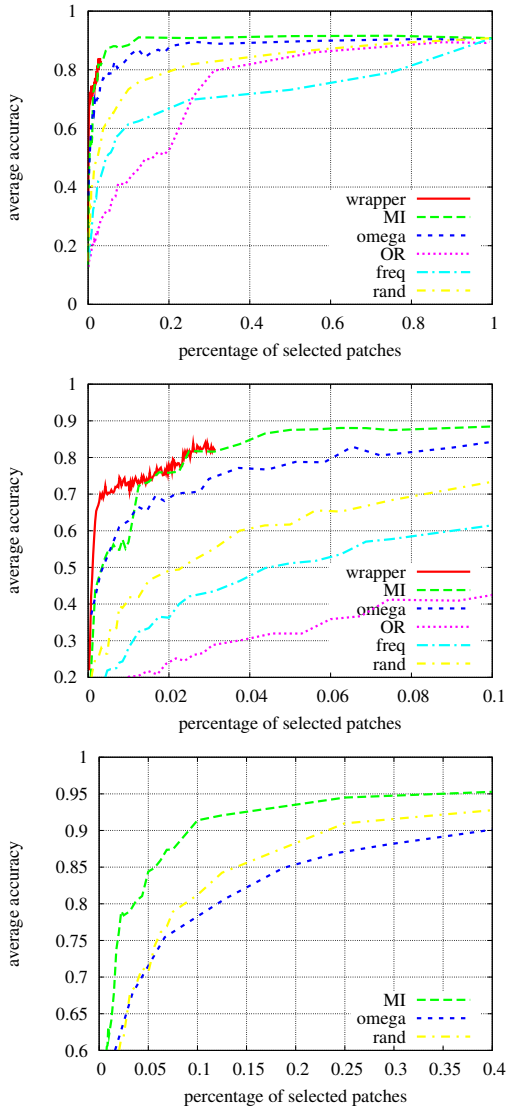


Figure 4: Influence of feature selection methods. Line 1 (2): visible database (zoomed). Line 3: IR database.

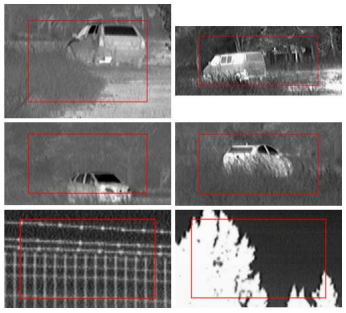

Figure 5: The regions analyzed by the framework are showed by the red rectangles.

| *Transformation* | *Performance* |
|---|---|
| **raw** | **83%** |
| ranging | 87% |
| proba | 78% |
| bin0 | 88% |
| binAuto | 92% |
| *Transformation* | *Performance* |
| **raw** | **89%** |
| ranging | 93% |
| bin0 | 96% |
| binAuto | 96% |

Table 1: The effects of data transformation. Top: visible dataset. Bottom: IR dataset

| rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Vis | 300 | 500 | 2500 | 2250 | 6000 | 200 | 150 |
| IR | 0 | 0 | 0 | 0 | 0 | 6 | 0 |

Table 2: The binarization thresholds automatically selected for the top 7 patches, with a MI ranking. Visible and Infrared databases

with 1% of the patches (i.e. 40), 4% with 3% of the patches and has the same accuracy with 5% of the patches. In the other database, `tree1` is 18% better than `1vs1` with 1% of the patches, 8% with 3% of the patches and has the same performance with 4% of of the patches.

We can also notice that the tree computation is crucial, because the random tree `tree2` is significantly worse than `tree1` (figure 6, last line).

From this observations we can conclude that HBC classifiers are more accurate with a small number of features.

## 4.5. Data transformation

In this section we want to observe the influence of data representation on the classification accuracy. We compare the following representations: `raw` (number of detections), `proba` (linear transformation by object), `ranging` (gives the same range to all dimensions), `bin0` (presence / absence of patch) and `binAuto` (optimal minimum detection number).

`Proba` and `ranging` are two popular transformation methods [9]. Compared to the original data (`raw`), we observe that `ranging` improves the performance and `proba` decreases it. The standard binarization `bin0` is also an improvement, since it outperforms `ranging`.

Our new data transformation method, `binAuto`, is the most efficient of the evaluated transformations. For the IR dataset, `binAuto` and `bin0` give the same improvement of accuracy: +7 points compared to `raw`. For the visible light dataset, `binAuto` is +9 points better than `raw`, and +4 points better than `bin0`. Figure 7 compares `bin0` and `binAuto` for the multi-class strategies `1vs1` and `tree`,

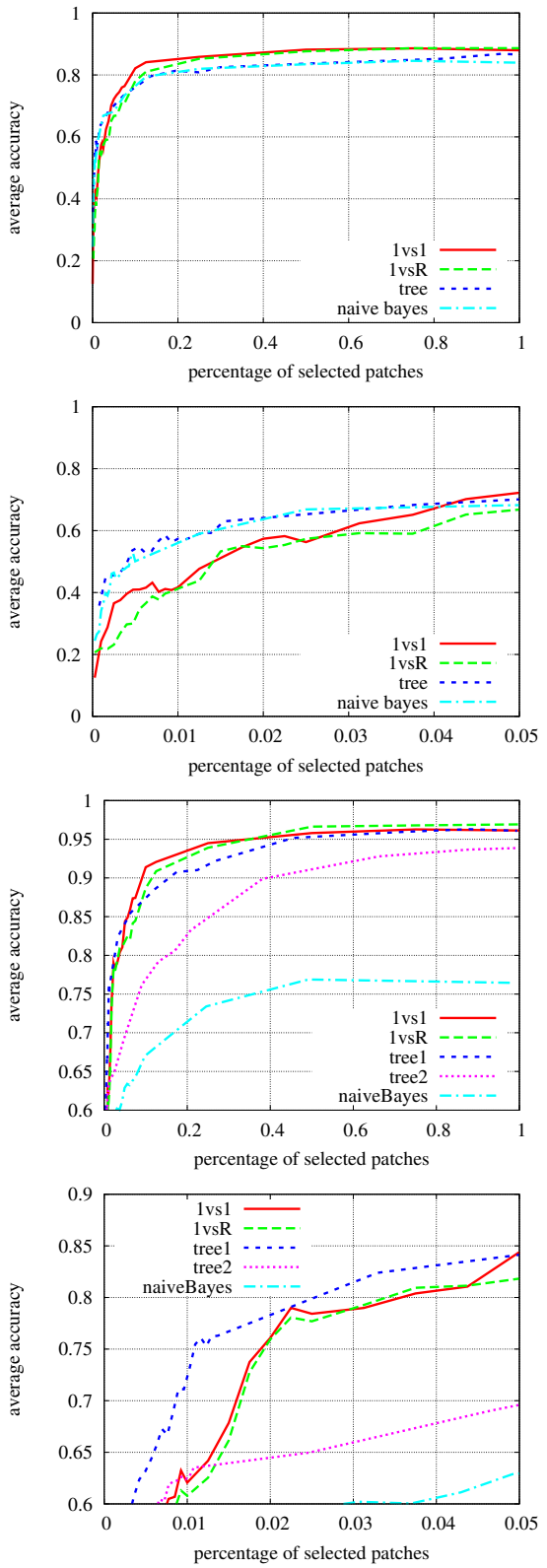


Figure 6: Multi-class strategies. Top: visible database (full graph and zoom). Bottom: IR database (full graph and zoom). The graphs show the mean accuracy in function of the percentage of selected patches.

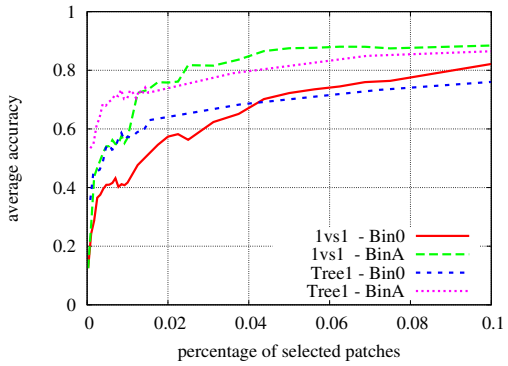


Figure 7: Automatic binarization increase the performance of `1vs1` and `tree`

in the visible database: in both cases, `binAuto` leads to a significant improvement of the accuracy.

We have to understand why `bin0` and `binAuto` have the same performance on the IR database, and different ones on the visible database. Table 2 shows the thresholds that were automatically selected for the two databases. The optimal thresholds are very high in the visible database — so `bin0` is less informative — whereas they are often zero in the IR database — so `bin0` is a good approximation of `binAuto`. Unlike `bin0`, the automatic binarization method can be used in many situations, because it does not require to set thresholds that are database dependent.

# 5. Conclusion

We have considered the problem of vehicle recognition for "bag of features" approach and the trade-off between the accuracy and the number of parts used in the model – directly affecting the computational speed. We proposed a new data representation, based on maximum mutual information binarization, that improves the classification accuracy. We also proposed a feature selection scheme adapted to hierarchical classifiers, which outperforms feature selection strategies applied to the standard `1vs1` SVM classifier when the number of parts selected are small. We achieved good results both for infrared vehicle categorization problem and for more generic visible object categorization problem.

Future works include the evaluation of wrapper methods for hierarchical classifiers, and the use of conditional mutual information to remove redundancy in the selected features.

# References


[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *PAMI*, 26(11):1475–1490, November 2004.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & sons, 1991.

[4] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *ICML*, pages 121–129, 1994.

[5] F. Jurie and W. Triggs. Creating efficient codebooks for visual recognition. In *ICCV05, to appear*.

[6] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *BMVC03*, 2003.

[7] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, June 2001.

[8] E. Haritaglu M. Betke and L. Davis. Multiple vehicle detection and tracking in hard real time. In *IEEE Intelligent Vehicles Symposium*, 1996.

[9] D. Mladenic and M. Grobelnik. Feature selection on hierarchy of web documents. *Decis. Support Syst.*, 35(1):45–87, 2003.

[10] S. Rajan and J. Ghosh. *An Empirical Comparison of Hierarchical vs. Two-Level Approaches to Multiclass Problems*. Springer, 2004.

[11] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR04*, pages II: 762–769, 2004.

[12] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[13] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, 1995.

[14] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *ICCV03*, pages 281–288, 2003.

[15] J. Willamowski, d. Arregui, G. Csurka, C. R. Dance, and L. Fan. Categorizing nine visual classes using local appearance descriptors. In *International Workshop on Learning for Adaptable Visual Systems (LAVS04), Cambridge, United Kingdom*, 2004.

[16] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.