Universität Hamburg

Technical Report

Decision trees for probabilistic top-down and bottom-up integration

Kasim Terzić Bernd Neumann

{terzic|neumann}@informatik.uni-hamburg.de

August 11, 2009

Zusammenfassung

Eine der Aufgaben, die Szeneninterpretationssysteme erledigen, ist die Zuordnung von mehrdeutigen Bildverarbeitungsergebnissen zu Konzepten aus einer Ontologie. Diese Entscheidungen sind in vielen Domänen unsicher und können durch Kontextinformationen verbessert werden. In diesem Bericht stellen wir einen Einsatz vor, wo die Klassenwahrscheinlichkeiten für Bildregionen mittels Entscheidungsbäumen geschätzt werden, und wir zeigen wie Kontextinformationen genutzt werden können, um die Klassifikationsrate zu verbessern.

Abstract

Scene interpretation systems need to match (often ambiguous) low-level input data to concepts from a high-level ontology. In many domains, these decisions are uncertain and benefit greatly from proper context. In this paper, we demonstrate the use of decision trees for estimating class probabilities for regions described by feature vectors, and show how context can be introduced in order to improve the matching performance.

1 Introduction

In the field of Computer Vision, there is a growing interest in using high-level knowledge for interpreting scenes from a wide range of domains. This involves vision tasks which go beyond single-object detection to provide an explanation of the observed scene. These tasks include inferring missing and occluded parts and recognising structure and relationships between objects in the scene. Typical examples include monitoring tasks such as airport activity recognition [8], interpreting building façades [12, 27, 25] or analysing traffic situations [20, 14].

As shown in [11], scene interpretation can be formally modelled as a knowledge-based process. Such a knowledge-based system, based on the configuration methodology, exists in the form of SCENIC [24]. The SCENIC system consists of a domain-specific knowledge base of concepts and an interpretation process which propagates constraints, instantiates concepts to instances, determines relations between instances, etc. Concepts are mainly aggregate models, their instances represent aggregate instantiations (or simply: "aggregates"), i.e. configurations of concrete objects in scenes. The interpretation process attempts to create a set of assertations about the scene that describe the observed evidence. The assertations describe instances of concepts from a domain-specific ontology.

The task of the middle layer of an interpretation system is then to match the detections from low-level image processing algorithms to concepts from the domain ontology. There are many examples in the literature where specific classes of objects are detected in the image with high accuracy [18, 19]. However, many domains exist where the classes have heterogeneous appearances and where there is considerable overlap between appearances, leading to many classification errors when using a purely bottom-up approach. An example is the domain of building façades which consists mostly of rectangular objects of varying sizes and considerable overlap between classes (see Figures 4 and 5). Previous research [6, 7] has shown that a purely appearance-based classification is difficult in this domain, even when it is reduced to a 4-class problem (e.g. *Roof, Sky, Ground* and *Façade*) . In domains like these, it may be preferable to explicitly model the uncertainty of classification so that high-level context can improve the decision.

This paper presents a multi-class classification scheme based on impure decision trees. A decision tree classifier is automatically learnt for a given combination of classes and feature vectors, and its leaves carry the class probabilities for given evidence for all the classes in the ontology. In other words, it serves as a discrete approximation of the conditional probability density functions $P(C|E)$ for all the classes. As such, it can express the uncertainty of classification decisions and can be easily combined with contextual priors (e.g. coming from high-level interpretation) for disambiguation. While impure decision trees are well-known, we are not aware of their use for scene interpretation.

The approach is evaluated on the eTRIMS database of annotated façade images. Three separate aspects of the decision trees are evaluated: bottom up classification in the façade domain compared to SVMs (Section 4.2), the accuracy of probability estimates (Section 4.3), and the effect of using contextual priors on the classification rate (Section 4.4). In this paper, we use manually updated priors in order to measure the effect that correct context has on the classification rate. The integration of automatically calculated priors is planned in the future.

In the following section, we introduce our domain and show why it makes classification difficult. In Section 3, we explain our classification methodology using decision trees. In Section 4, we evaluate the performance on an annotated image database. Section 5 summarises our findings and discusses future work.

2 The Facade Domain

Recently, there has been an increased interest in interpreting building scenes, e.g. for localisation [23], vehicle navigation [14] and photogrammetry [17]. Buildings, being man-made structures, exhibit a lot of regularity that can be exploited by a reasoning system, but there is still enough variety within this structure to present a challenge for interpretation and learning tasks [9, 10].

A large database of annotated façade images exists as an outcome of the eTRIMS project [16], which can serve both as ground truth for classification and interpretation tasks, and as learning data. It contains close to 1000 fully annotated images. A sample image from the database is shown in Figure 1. The high-level ontology describing the domain used for the experiments in this paper contains the following classes: *Balcony, Building, Canopy, Car, Cornice, Chimney, Door, Dormer, Entrance, Façade, Gate, Ground, Pavement, Person, Railing, Road, Roof, Sign, Sky, Stairs, Vegetation, Wall, Window,* and *Window Array.* Some of these classes represent

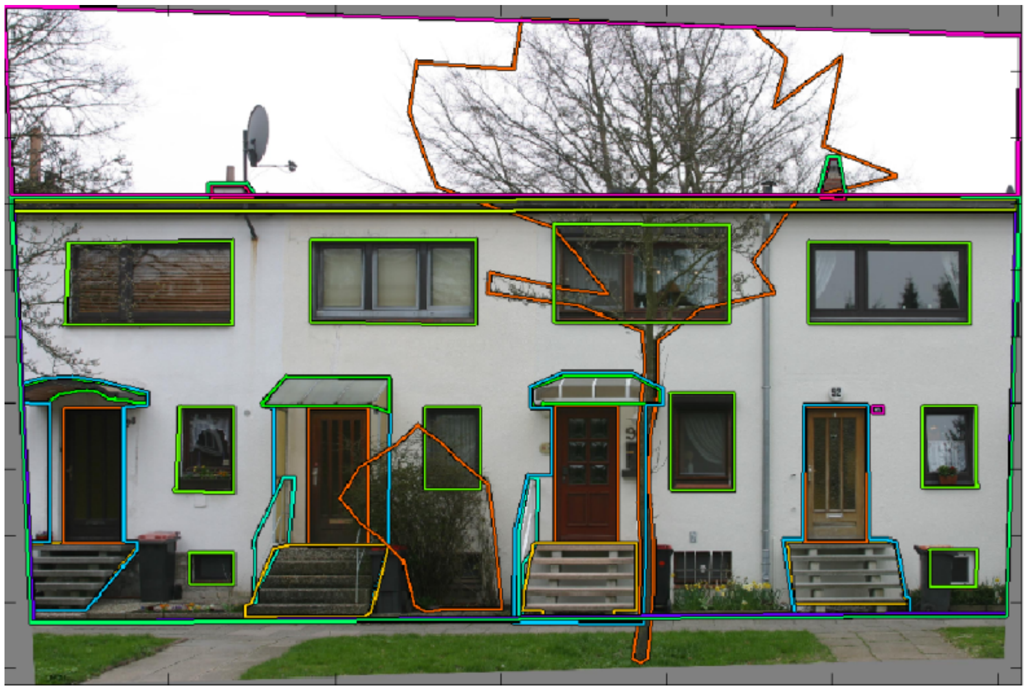

Figure 1: Example from the eTRIMS annotated façade database. Each object is marked by a bounding polygon and a label from the ontology (indicated here by different colours.)

primitive objects without parts (like *Door* or *Window*), and some represent aggregates consisting of primitive objects (like *Balcony* or *Window Array*) or other aggregates (like *Façade*), thus forming a hierarchical structure.

For several reasons, the façade domain presents a number of challenges regarding classification:

- Most of the objects are rectangular (façades, windows, doors, railings, etc.) and of similar size. Some of the objects can come in virtually any colour (walls, doors, cars), some are semi-transparent (railings, vegetation) and blend with the objects behind them, and some can reflect other objects (windows and window panes). This leads to significant overlap between classes for most feature descriptors.
- The variability of appearance within each class is greater than the difference between classes, making it difficult to create compact appearance models.

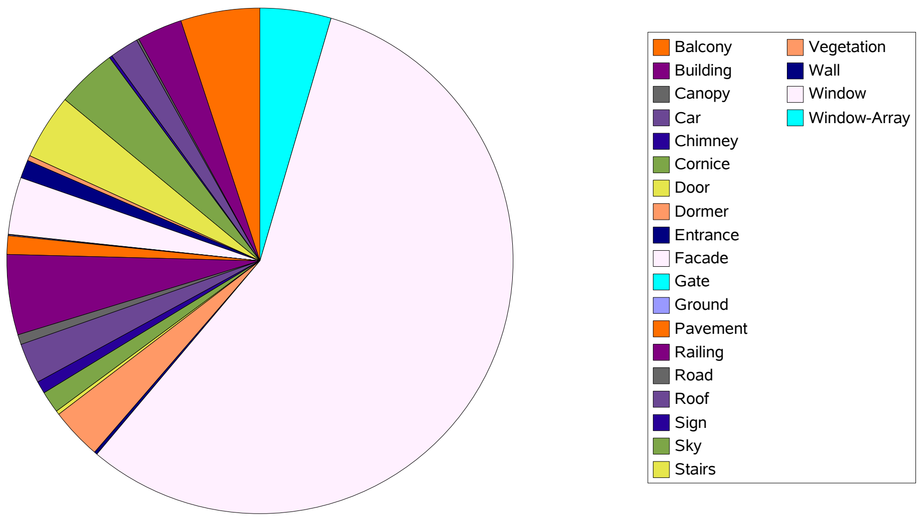

Figure 2: Distribution of object classes. Windows have by far the largest prior, followed by window arrays, railings, balconies and doors.

- Some aggregate classes consist of parts in a loose configuration and as such don't have a characteristic appearance by themselves, e.g. balcony or façade.
- Some classes are distinctly more common than others. As shown in Figure 2, around 55% of all annotated objects in the eTRIMS database are windows. Thus, a classifier seeking to minimise the expected total error will tend to misclassify objects as windows.

The difficulty of bottom-up classification in the façade domain was also discussed in [6] and [2]. On the other hand, the façade domain provides a lot of context which can be useful for classification. To name a few examples, entrances are usually located at the bottom of a façade, roofs at the top, windows occur in arrays, etc.

There are several approaches which exploit this context in the façade domain,

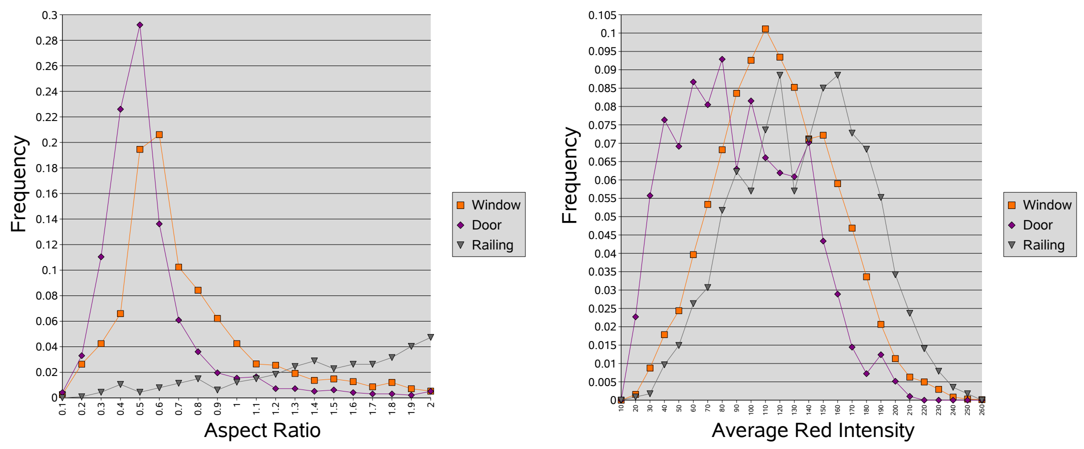

Figure 3: Distribution of the aspect ratio (left) and average intensity of the red channel (right) for three common classes. It can be seen that making certain decisions is difficult even for the three-class case.

using configuration [13], Markov Random Fields [10] or grammars [25] . Our approach uses a probabilistic model for context generation in terms of dynamic priors from a Bayesian Compositional Hierarchy (BCH) [21]. A BCH is a special kind of Bayesian Network with aggregates as nodes and isomorphic to the aggregate hierarchy. Dynamic priors are provided by propagating the effect of evidence assignments to other nodes of the BCH. Details of high-level interpretation, however, will not be provided in this report, which focusses on the low-level stage using decision trees.

3 Learning Decision Trees

A decision tree is a tree where the leaf nodes represent classifications and each non-leaf node represents a decision rule acting on an attribute of the input sample. A sample described by a d-dimensional feature vector f and consisting of d scalar attributes is classified by evaluating the decision rule at the root node and passing the sample down to one of the subnodes depending on the result, until a leaf node is reached. The result is a partitioning of the feature space into labelled disjoint regions.

In a binary decision tree, the rules correspond to yes/no questions and each nonleaf node has exactly two children. The most common type of decision trees, called *univariate* decision trees, only act on one dimension at a time

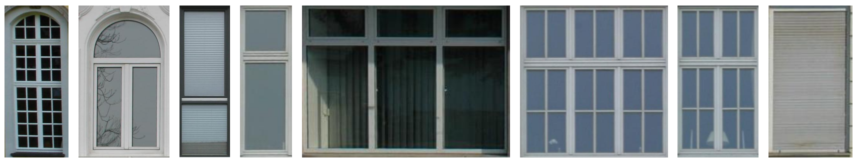

Figure 4: Some windows from the annotated database.

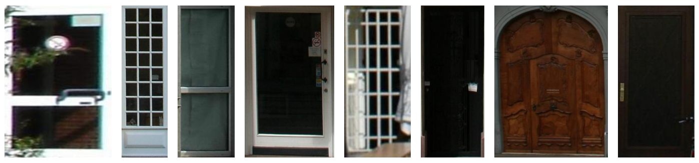

Figure 5: Some doors from the annotated database. The appearance and shape varies a lot and there is significant overlap with the Window class.

and thus result in an axis-parallel partitioning. The experiments described in this paper used univariate decision trees.

If the leaves are allowed to correspond to more than one class, they are called impure leaves. If each class in a leaf node is given a probability, such trees can be used to model the uncertainty of the classification result. Essentially, they provide a discrete approximation of a probability density function, where the discretisation can be irregular.

For many problems, decision trees have competitive performance compared to other classification schemes [26]. At the same time, they have the advantage of having a result that can be understood intuitively because they split the feature space into regions with axis-parallel boundaries. In addition to providing bottom-up classification of low-level evidence, they can also *describe* the visual appearance of high-level concepts by specifying a region of feature space. This is an appealing property for scene interpretation, because it simplifies top-down processing by making it possible to pass expectations of low-level appearance of expected objects to image processing algorithms in a compact and understandable form.

3.1 Learning algorithm

We now address some aspects of decision tree learning. Since the space of all possible trees is huge, greedy algorithms for learning the best tree are usually employed. The tree starts as a single root node containing all the samples and is recursively subdivided according to a splitting criterion. There are many splitting criteria in use for learning decision trees, two of the most popular are information content and the Gini coefficient[1], used in our experiments. For both criteria, we need to know the conditional probability $P(\omega|t)$ at a node t, which can be approximated as

$$P(\omega_i|t) = \frac{N_i(t)}{N(t)}$$

where $N_i(t)$ is the number of samples in t belonging to class ω_i, and $N(t)$ is the number of all samples in t,

The information content $I(t)$ of a node t is the number of bits needed to encode the information of that node. Given m classes, the information content of an impure node is given as

$$I(t) = I(P(\omega_1|t), \ldots, P(\omega_m|t)) = \sum_{i=1}^{m} -P(\omega_i|t) log_2 P(\omega_i|t) \tag{1}$$

For a binary decision with equal probabilities, the information content is $I(\frac{1}{2}, \frac{1}{2}) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$bit. For a biased decision where one choice has a 99% probability, it is $I(\frac{1}{100}, \frac{99}{100}) = 0.08$ bits. The goal of decision tree learning is to minimise the uncertainty with every step, and since uncertain decisions are encoded with more information, splits which minimise the information content of the new trees should be favoured.

The Gini coefficient is a measure of the impurity of a node, and as such also related to the information content of the node. The Gini coefficient of node t is defined as

$$G(t) = \sum_{i \neq j} P(\omega_i|t)P(\omega_j|t) \tag{2}$$

For a given split that divides t into t_l and t_r, the change in the Gini coefficient is given as

$$\Delta G(sp, t) = G(t) - (G(t_l)P_l + G(t_r)P_r)$$

where P_l and P_r are the priors for the left and right subnode, respectively. The best split is the one that maximises $\Delta G(sp, t)$.

[1]More detailed explanations can be found in [22] and [26]

When learning a decision tree, the node with the highest impurity (measured as high information content or high Gini coefficient) is split in a way that maximises the splitting criterion. This entails two decisions: choosing the attribute (dimension) to split on and choosing its best value for the split. A simple approach, used for learning the trees described in this paper, is to perform an exhaustive search through the space of all possible splits in all possible dimensions, and to choose the one that minimises the Gini coefficient of the resulting sub-nodes. Given a set of n samples, each described by a d-dimensional feature vector f, an exhaustive search of all possible splits in each dimension has a complexity of $O(nd)$.

3.2 Pruning

Overfitting is a well-known problem with learning of decision trees [22, 26]. The leaf-splitting can be continued until all leaves have pure class membership. Such learnt trees describe the training set well, but if the data are not perfectly separable or contain noise, they do not generalise well to unseen examples, essentially modelling the noise in the training set. One can terminate the learning once a stopping criterion is fulfilled (e.g. minimum change of impurity function), or use one of many pruning algorithms to reduce the maximal tree.

Classification and Regression Trees (CART) were first introduced by Breiman [3] and still present a popular method for pruning learnt decision trees. The basic idea is to add a constant α to the impurity measure at each split, as a measure of the cost of additional complexity introduced by the split.

More specifically, if $R(t)$ is the measure of impurity at node t (the misclassification rate), then $R_\alpha(t) = R(t) + \alpha$ is the complexity measure of the node t. If $\tilde{T}$ is the set of all leaves in a tree T, and $|\tilde{T}|$ the cardinality of $\tilde{T}$, then $R(T) = \sum_{t \in \tilde{T}} R(t)$ is the estimated misclassification rate of a tree T, and

$$R_\alpha(T) = \sum_{t \in \tilde{T}} R_\alpha(t) = R(T) + \alpha|\tilde{T}|$$

is the estimated complexity-misclassification rate of T. If we define T_t to be a subtree with node t at its root, we can calculate the strength of the link from node t to its leaves as

$$g(t) = \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1} \tag{3}$$

The nodes with a low $g(t)$ are punished as they add complexity without significantly improving the classification result. The algorithm starts with the maximal tree and calculates $g(t)$ for all nodes. The node with the lowest value of $g(t)$ is made into a leaf, and all of its children are removed. The new values for $g(t)$ are calculated for all the predecessors of the affected node, and the process is repeated on the new tree.

The result is a succession of trees, starting with the initial, maximal tree, and ending with a tree containing only the root node. Each of these trees is a classifier. All the trees are tested on an unseen validation dataset and the tree with the best classification rate is selected as the final classifier.

3.3 Classification

If an impure leaf l contains the samples of several classes, an estimate of $P(C|L)$ for all classes and leaves can be formulated as

$$P(c|l) = \frac{N_c(l)}{N(l)}$$

where $N_c(l)$ is the number of samples in l belonging to class c and $N(l)$ is the number of all samples in l. The probabilities at the leaves $P(C|L)$ reflect the success rate achieved with the training set used to learn the tree.

Instead of encoding $P(C|L)$, we can observe how often an object belonging to class c generates evidence described by the leaf l, giving the class-conditional probability $P(L|C)$ for all classes and leaves. Then, Bayes rule gives the posterior probability as

$$P(C|L) = \frac{P(L|C)P(C)}{P(L)} \tag{4}$$

Finding the class for which $P(C|L)$ is maximum gives a MAP classifier. Since each evidence sample e is mapped into a leaf l of the decision tree, $P(C|L)$ serves as a discrete approximation of $P(C|E)$.

3.4 Incorporating Context

The formulation in Equation 4 allows for introducing updated priors $P'(C)$, which reflect the scene context coming from a high-level reasoning system or an additional knowledge source. If we assume that the typical appearance of the classes is not affected by context, i.e. that $P(L|C) = P'(L|C)$, the probability that a leaf l belongs to class c can be written as

$$P'(C|L) = \frac{P(L|C)P'(C)}{P'(L)} = \frac{P(L|C)P'(C)}{\sum_c P(L|C)P'(C)} \tag{5}$$

The leaves of a decision tree typically store $P(C|L)$ and not $P(L|C)$, so we derive an update mechanism which calculates $P'(C|L)$ from $P(C|L)$, $P(C)$ and $P'(C)$.

$$\begin{aligned} P'(C|L) &= \frac{P'(CL)}{P'(L)} = \frac{P(L|C)P'(C)}{P'(L)} = \frac{P(L|C)P'(C)}{P'(L)} \\ &= \frac{P(LC)\frac{P'(C)}{P(C)}}{P'(L)} = \frac{P(C|L)P(L)\frac{P'(C)}{P(C)}}{P'(L)} \end{aligned} \tag{6}$$

$P'(L)$ can be expressed as

$$\begin{aligned} P'(L) &= \sum_c P'(CL) = \sum_c P'(L|C)P'(C) = \sum_c P(L|C)P'(C) \\ &= \sum_c P(LC)\frac{P'C)}{P(C)} = \sum_c P(C|L)P(L)\frac{P'(C)}{P(C)} \\ &= P(L)\sum_c P(C|L)\frac{P'(C)}{P(C)} \end{aligned} \tag{7}$$

Inserting 7 into 6 gives

$$P'(C|L) = \frac{P(C|L)\frac{P'(C)}{P(C)}}{\sum_c P(C|L)\frac{P'(C)}{P(C)}} \tag{8}$$

where $P(C)$ are the domain priors of the training set used for learning the tree, $P(C|L)$ are the conditional class probabilities at the leaves of the decision tree, and $P'(C)$ are the updated class priors.

Table 1: The means of the Gaussian distributions used to generate synthetic samples.

$\mu(class1)$	-0.5	-0.5	-0.5
$\mu(class2)$	0.5	0.5	0.5
$\mu(class3)$	-0.5	-0.5	0.5
$\mu(class4)$	0.5	0.5	-0.5
$\mu(class5)$	-0.5	0.5	-0.5
$\mu(class6)$	-0.5	0.5	0.5
$\mu(class7)$	0.5	-0.5	0.5
$\mu(class8)$	0.5	-0.5	-0.5

4 Evaluation

Our decision trees were tested on both synthetic data and annotated objects from the façade domain. In each case, we tested automatically learnt trees as pure bottom-up classifiers, and then evaluated the effect of updated context-priors on the classification rate.

4.1 Synthetic Data

We first tested the decision trees on synthetic 4-class and 8-class data. Each sample is drawn from a 3-dimensional Gaussian distribution. Table 1 shows the means of the distributions, and Table 2 shows the standard deviations and the priors of the classes. The first dataset has four equally probable classes, the second set has classes chosen to be more similar to the façade domain, and the third set increases the standard deviation leading to more overlap between classes. Datasets 4 to 6 follow the same pattern, using 8 classes.

The results were compared with SVM-based multiclass classifiers using the svmlight software [15]. For each N-class dataset, we obtained three results: using the learnt decision tree followed by a MAP classification, by choosing the strongest response from N one-against-all SVM classifiers (which we refer to as *SVM1*), and finally by performing a majority vote among $N(N-1)/2$ pairwise SVM classifiers (referred to as *SVM2*). The 4-class datasets were tested using 10000 training samples (of which 1000 are used for pruning

Table 2: The priors on the classes of the synthetic data and the standard deviation used for the Gaussian distributions modelling the classes.

	DS 1	DS 2	DS 3	DS 4	DS 5	DS 6
P(class1)	0.25	0.1	0.1	0.125	0.05	0.05
P(class2)	0.25	0.1	0.1	0.125	0.05	0.05
P(class3)	0.25	0.2	0.2	0.125	0.05	0.05
P(class4)	0.25	0.6	0.6	0.125	0.05	0.05
P(class5)	0	0	0	0.125	0.05	0.05
P(class6)	0	0	0	0.125	0.10	0.10
P(class7)	0	0	0	0.125	0.10	0.10
P(class8)	0	0	0	0.125	0.55	0.55
σ_{1-8}	0.5	0.5	1.5	0.5	0.5	1.5

Table 3: Comparison of a one-against-all SVM classifier (SVM1), a pairwise SVM classifier (SVM2), and our decision tree on 6 synthetic datasets. The numbers represent the classification rate for a given dataset.

	SVM1	SVM2	Decision tree	
DS1	0.7805	0.7801	0.7654	(73 nodes)
DS2	0.8395	0.8412	0.8311	(109 nodes)
DS3	0.6843	0.6863	0.6745	(119 nodes)
DS4	0.5825	0.5915	0.5829	(411 nodes)
DS5	0.7238	0.7236	0.7145	(335 nodes)
DS6	0.5604	0.5793	0.5722	(25 nodes)

the trees) and 10000 test samples. The 8-class datasets used 20000 training samples (2000 for pruning) and 20000 test samples.

The results, shown in Table 3 show the comparison of our approach with the SVM-based classifiers. The performance is within a percentage point of the SVM classifiers in almost all cases, outperforming one of the SVM classifiers on datasets 4 and 6.

Table 4: The composition of the feature vector.

f_0	area
f_1	compactness: $4\pi \times area/perimeter^2$
f_2	aspect ratio: $width/height$
f_3	rectangularity: $area/(width \times height)$
f_{4-5}	mean and standard deviation of the red channel
f_{6-7}	mean and standard deviation of the blue channel
f_{8-9}	mean and standard deviation of the green channel
f_{10-18}	8-bin edge orientation histogram

4.2 Real Data

Our experiments on real data are based on the annotated façade image database from the eTRIMS project. All images are fully annotated using bounding polygons and class labels from a common ontology. For the experiments in this paper, we used 599 rectified images (façade edges are parallel to the image axes), consisting of 27922 objects in total. From these images, we used 15357 training objects, 6981 validation objects used for pruning, and 5584 testing objects. We use rectified images .

Table 4 shows the composition of the 18-dimensional feature vector used to describe each object. We use simple and general features, because previous work on feature selection showed these features to be useful in the façade domain [6, 7], and more complex features such as statistical moments and colour histograms did not perform as well in our experiments.

The results of the decision tree classifier learnt for the 24-class façade object problem using the Gini coefficient for optimisation can be seen in Figure 6. The overall classification rate across all classes is 75.63%, with most classes showing a strong peak at the diagonal of the confusion matrix (see Figure 6).

It is apparent that the classes *Facade* and *Building* are often confused, as are *Road* and *Pavement*, but this is an expected result, given how visually similar these classes often are. This is a point where high-level context (in terms of a prior expectation for the classes) could improve the classification results.

Another interesting result is the poor performance with classes *Sign, Chimney* and *Door*. In the case of *Sign* and *Chimney*, the prior of the classes is so low that classifying all of them as windows actually reduces the overall error

Table 5: Comparison of a one-against-all SVM classifier (SVM1), a pairwise SVM classifier (SVM2), and our decision tree on 5584 objects from 599 annotated images from the façade domain.

SVM1	SVM2	Decision tree	
0.7092	0.6999	0.7563	(601 nodes)

rate. The prior of the class *Door* is quite high but, as shown in Figure 5, the visual appearance is often very close to the appearance of the *Window* class, which has a far higher prior. The solution to these problems is to introduce contextual information in the form of updated priors for different image regions. If there is a strong scene context suggesting one class over the other, this can be used for disambiguation, as will be shown in Section 4.4.

Once again, the results were compared with SVM-based multiclass classifiers using the svmlight software. We used two SVM-based classifiers: based on 24 one-against-all SVM classifiers (SVM1), and based on 276 pairwise SVM classifiers (SVM2). All three tests were performed on exactly the same objects, using the same features to keep results comparable. The only difference was that all individual features were scaled to between 0 and 1 for the SVMs. Since SVMs do not need a validation set, the objects used for pruning the decision tree were used as additional training objects for the SVMs. We used the default kernel (radial basis function) and default parameters (determined automatically by the svmlight software).

Table 5 shows the results. Our decision-tree based method outperforms both SVM-based methods in bottom-up classification. The confusion matrices for the SVM-bases classifiers are shown in Figures 7 and 8. Another interesting observation is that the problems with classification of doors are even more pronounced when using SVM-based classifiers, as opposed to decision trees. The performance on the *Balcony* class is also worse.

	Balcony	Building	Canopy	Car	Chimney	Cornice	Door	Dormer	Entrance	Facade	Gate	Ground	Pavement	Person	Railing	Road	Roof	Sign	Sky	Stairs	Vegetation	Wall	Window	Window-Array
Balcony	106	2	0	1	0	0	0	0	5	2	0	0	1	0	5	0	16	0	0	0	7	0	77	9
Building	9	79	0	1	0	0	0	0	0	72	0	0	0	0	1	0	11	0	1	0	6	0	9	5
Canopy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	0	0	1	0	2	1
Car	1	0	0	53	0	0	2	4	4	1	0	0	1	0	7	2	10	0	0	0	13	0	23	0
Chimney	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	13	0
Cornice	0	0	0	0	0	192	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	1	0
Door	0	0	0	0	0	0	49	0	1	0	0	0	0	0	0	0	0	0	0	0	6	0	149	2
Dormer	5	0	0	1	0	0	0	1	0	0	0	0	0	0	3	0	1	0	0	0	0	0	5	1
Entrance	3	0	0	1	0	0	4	0	14	4	0	0	0	0	2	0	3	0	0	0	2	0	40	4
Facade	8	43	0	0	0	0	1	1	0	145	0	0	0	0	1	0	5	0	0	0	3	0	10	6
Gate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Ground	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0	0	0	0	0
Pavement	4	0	0	4	0	0	0	0	0	1	0	0	15	0	9	11	10	0	0	1	8	0	11	2
Person	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Railing	9	0	1	4	0	3	0	1	0	0	0	0	2	0	137	1	5	2	0	0	7	0	51	9
Road	1	0	0	1	0	0	0	1	0	1	0	0	11	0	2	10	4	0	1	0	9	0	0	3
Roof	23	3	0	10	0	0	0	1	6	0	0	0	7	0	5	0	76	0	0	0	19	0	14	5
Sign	0	0	0	1	0	2	2	0	0	0	0	0	2	0	8	0	0	5	0	0	2	0	25	0
Sky	2	1	0	1	0	0	0	1	0	2	0	0	0	0	0	1	7	0	90	0	3	0	3	0
Stairs	2	0	0	2	0	0	0	1	0	0	0	0	0	0	3	0	1	0	0	1	1	0	6	0
Vegetation	6	8	0	7	0	0	0	0	4	2	0	0	7	0	7	2	21	0	1	0	102	0	22	3
Wall	0	1	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Window	34	3	0	3	0	1	39	0	2	0	0	0	1	0	28	0	5	1	1	1	8	0	2976	17
Window-Array	4	0	1	1	0	1	0	0	0	14	0	0	1	0	21	0	1	1	0	0	2	0	54	172

Figure 6: Confusion matrix for the learnt decision tree. Overall classification rate is 75.63%.

	Balcony	Building	Canopy	Car	Cornice	Chimney	Door	Dormer	Entrance	Facade	Gate	Ground	Pavement	Person	Railing	Road	Roof	Sign	Sky	Stairs	Vegetation	Wall	Window	Window-Array
Balcony	28	2	0	1	2	1	0	1	4	3	0	0	0	0	5	0	4	0	0	1	9	0	166	4
Building	2	92	0	0	0	0	0	0	1	49	0	0	0	0	1	0	4	0	1	0	23	0	17	4
Canopy	0	0	0	0	2	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	4	2
Car	4	1	0	47	0	0	0	0	0	0	0	0	0	0	0	2	8	0	0	0	10	0	50	0
Cornice	0	0	0	0	160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	1
Chimney	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
Door	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	200	0
Dormer	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	16	0
Entrance	2	1	0	1	0	0	0	0	3	1	0	0	0	0	0	0	0	0	0	0	12	0	57	0
Facade	2	45	1	2	0	0	0	0	1	120	0	0	0	0	2	0	2	0	0	0	16	0	25	7
Gate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Ground	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0
Pavement	0	1	0	4	2	0	0	0	0	1	0	0	16	1	6	9	6	1	2	0	8	0	19	0
Person	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Railing	2	0	1	0	13	0	0	0	0	1	0	0	0	1	30	1	9	0	1	0	6	0	165	2
Road	0	1	0	3	1	0	0	0	0	3	0	0	9	0	0	8	4	0	0	0	10	0	3	2
Roof	2	21	0	6	4	0	0	0	0	3	0	0	0	0	1	0	59	0	0	1	34	0	36	2
Sign	0	0	0	4	2	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	38	1
Sky	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	105	0	1	0	4	0
Stairs	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	13	0
Vegetation	2	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	10	1	0	0	151	0	26	0
Wall	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Window	5	2	0	0	16	0	0	0	1	0	0	0	1	1	2	0	4	0	3	0	14	0	3070	1
Window-Array	0	1	0	0	14	0	1	0	0	16	0	0	0	0	1	0	0	0	0	0	1	0	169	70

Figure 7: Confusion matrix for the one-against-all SVM classifier (SVM1). Overall classification rate is 70.92%.

	Balcony	Building	Canopy	Car	Cornice	Chimney	Door	Dormer	Entrance	Facade	Gate	Ground	Pavement	Person	Railing	Road	Roof	Sign	Sky	Stairs	Vegetation	Wall	Window	Window-Array
Balcony	3	2	0	2	0	0	0	0	0	2	0	0	0	0	0	0	7	0	0	0	4	0	209	3
Building	6	84	0	1	0	0	0	0	0	42	0	0	0	0	0	0	12	0	0	0	13	0	32	4
Canopy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	5	4
Car	0	1	0	47	0	0	0	0	0	0	0	0	1	0	0	0	6	0	0	0	12	0	54	0
Cornice	0	0	0	0	158	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	3
Chimney	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0
Door	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	202	0
Dormer	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0
Entrance	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	8	0	64	0
Facade	2	42	0	0	0	0	0	0	0	121	0	0	1	0	0	0	8	0	0	0	5	0	36	8
Gate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Ground	0	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
Pavement	0	0	0	1	1	0	0	0	0	1	0	0	16	0	0	9	8	0	0	0	5	0	35	0
Person	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Railing	0	0	0	0	6	0	0	0	0	1	0	0	1	0	0	0	8	0	0	0	3	0	210	3
Road	0	3	0	2	0	0	2	0	0	1	0	0	13	0	0	6	1	0	0	0	6	0	9	1
Roof	5	14	0	5	0	0	0	0	0	1	0	0	5	0	0	0	62	0	0	0	22	0	50	5
Sign	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	40	1
Sky	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	103	0	1	0	4	0
Stairs	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	14	0
Vegetation	2	7	0	5	0	0	0	0	0	0	0	0	0	0	0	3	16	0	0	0	122	0	37	0
Wall	0	1	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Window	0	3	0	2	14	0	0	0	0	0	0	0	1	0	0	3	4	0	0	0	10	0	3079	4
Window-Array	0	2	0	0	2	0	0	0	0	5	0	0	2	0	0	0	1	0	0	0	0	0	154	107

Figure 8: Confusion matrix for the pairwise SVM classifier (SVM2). Overall classification rate is 69.99%.

Table 6: Comparison of estimated class probabilities for the strongest class with the actual classification rate. Leaves with similar probabilities for the strongest class were grouped together in bins. The left column shows the expected result (mean value of each bin), and the other columns show the actually measured classification rates for these leaves. The best probability estimates were observed with a smoothed tree and m=1. In one case, no leaves had a probability estimate in the given range, this is indicated as ”n/a”.

Expected	No smoothing	m=0.1	m=0.5	**m=1**	m=5	m=10
0.95	0.92	0.94	0.94	**0.95**	0.95	0.95
0.85	0.81	0.84	0.85	**0.87**	0.89	n/a
0.75	0.65	0.67	0.74	**0.75**	0.82	0.88
0.65	0.49	0.50	0.62	**0.64**	0.72	0.71
0.55	0.45	0.44	0.50	**0.60**	0.67	0.54
0.45	0.36	0.36	0.43	**0.42**	0.45	0.27

4.3 Accuracy of probability estimates

One nice property of decision trees is that they provide an estimate of the probability of correct classification (without consideration of context). In scene interpretation systems, this is useful information since it can be used to influence the order of interpretation steps. However, it is well-known that when trees are learnt in a way that tries to maximise the classification rate, the probability estimates are incorrect, especially for domains with unbalanced priors [28].

Several *probability smoothing* approaches have been proposed in the literature to address this problem [1, 4, 28]. A common and effective smoothing approach is m-estimation introduced by Cestnik [5]. The probability estimate at the leaves $P(c|l) = \frac{N_c(l)}{N(l)}$ is replaced by $P_s(c|l) = \frac{N_c(l)+P_d(c)m}{N(l)+m}$, where $P_d(c)$ is the domain prior for class c. We calculated the smoothed probabilities $P_s(C|L)$ for all classes and leaves. Since m-smoothing is a heuristic which affects different classes differently, we finally renormalised all probabilities in all leaves so they sum to one again. The parameter m determines how strongly the probabilities at the leaves are adjusted towards the domain prior. We determined the parameter m experimentally, as described below.

We compared the estimated probability provided by the leaves of the learnt

Table 7: Comparison of classification rate for the original tree (left column) and trees smoothed with different values of m (right).

No smoothing	m=0.1	m=0.5	m=1	m=5	m=10
0.7563	0.754835	0.752507	0.750895	0.708453	0.682307

decision trees with the actual classification rate. To this end, we compared a tree with no smoothing to a number of trees corresponding to different values for the parameter m. Ideally, the probability estimate of the decision tree will be the same as the probability observed in practice. In other words, if an object is classified as a window with $P(window|l) = 0.7$, we expect that such a classification will be correct in 70% of the cases. In order to test this, we have grouped together nodes with similar $P(c_{strongest}|l)$ and measured the actual classification rate for each group.

Table 6 summarises the results. We show the original tree (no smoothing) and smoothed trees using $m = 0.1$, 0.5, 1, 5 and 10. It can be seen that smoothing improves the probability estimates, and that the best results were achieved with m=1. One downside of smoothing is that it usually reduces classification accuracy. The effect of different smoothing factors on the classification rate is shown in Table 7. It can be seen that smoothing with m=1 doesn't impact classification rate strongly, and still significantly improves the probability estimates, making it the best choice for this domain.

4.4 Contextual information

We have tested the effect that changing the class priors has on the classification rate. We have simulated correct scene context by artificially altering the priors $P(C)$. For each tested object, we set the prior on the correct class to a certain value $P'(C)$ and renormalised all other priors so they sum all up to one again.

Figure 9 shows the effect of updated $P'(C)$ on the overall classification rate. It can be seen that even small changes to the prior can have a great effect on the overall classification rate. As the prior for the correct class approaches one, the overall error tends towards zero, of course.

Figure 10 shows the effect on three different classes from the façade domain. The *Window* class has a very high domain prior (around 55%), the *Stairs*

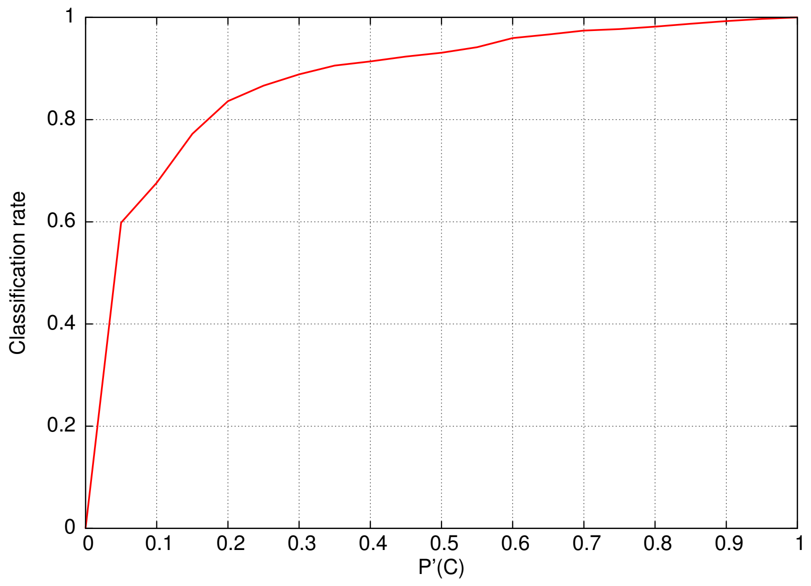

Figure 9: The effect of updated prior $P'(C)$ on the overall classification rate.

class has a very low domain prior (around 0.3%), and the *Door* class is relatively common (around 4%), but easily confused with the *Window* class. The graphs show that context is particularly helpful for less common and easily confused classes.

The context was simulated in these experiments, but it can be replaced by dynamic priors from Bayesian Compositional Hierarchies [21] or a similar probabilistic reasoning scheme in the future. The improvements shown in Figures 9 and 10 suggest that scene context in the form of updated priors will lead to improved classification.

5 Summary and Future Work

We have shown the application of decision trees to uncertain classification in a complex, multi-class domain. Decision trees offer competitive performance to standard multi-class SVM classification schemes on synthetic data, and better performance on the façade domain. At the same time, they allow easy incorporation of context in the form of class priors.

Currently, work is underway to integrate this middle-level classification frame-

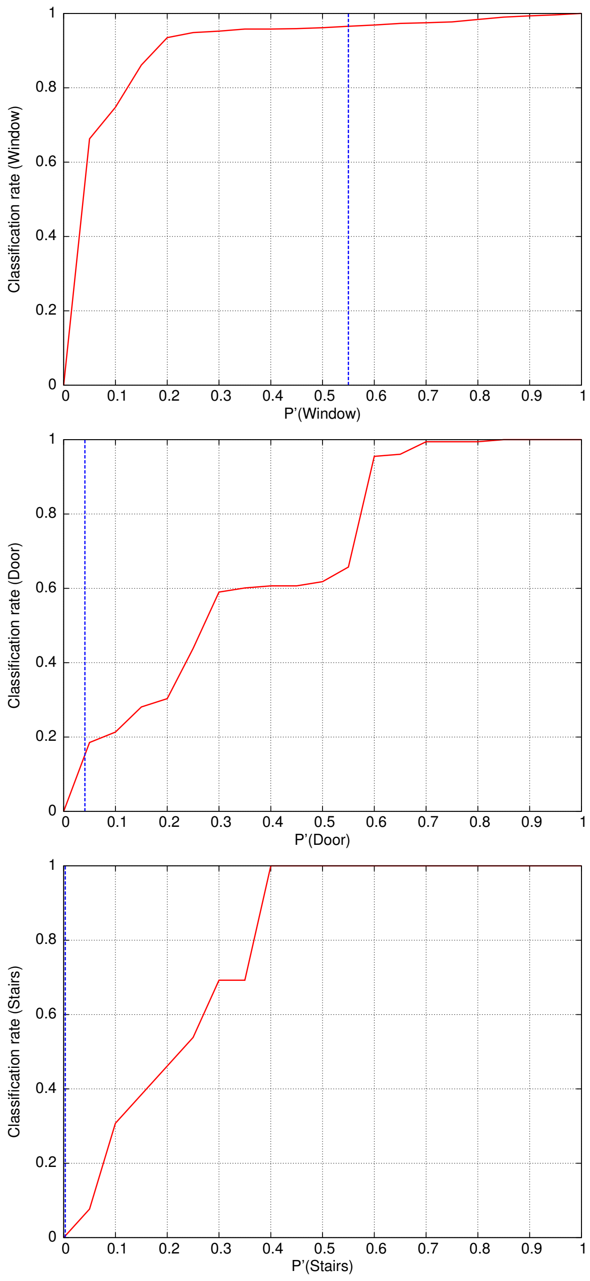

Figure 10: The effect of updated prior $P'(C)$ on three classes from the façade domain. From top to bottom, they are: *Window, Door* and *Stairs*. The vertical blue line shows the classification using the domain prior $P(C)$ without any change. The curves for door and stairs are jagged because they were obtained from fewer samples, which were represented by fewer nodes.

work into the scene interpretation system *SCENIC*. Also, the use of dynamic priors provided by a Bayes Compositional Hierarchy (BCH) is being investigated. These context-specific priors on classes in a scene should improve the classification results, especially for visually similar and often-confused classes.

An interesting extension of this work is the feedback to image-processing algorithms. Decision trees offer a partitioning of the feature space into axis-parallel, easy-to-describe blocks. Given a strong expectation for a certain class of an object, it is possible to formulate a description of the object in terms of allowed feature ranges, which can help a low-level algorithm detect it.

Acknowledgement

This research has been supported by the European Community under the grant IST 027113, eTRIMS - eTraining for Interpreting Images of Man-Made Scenes.

References

[1] Lalit R. Bahl, Peter F. Brown, Peter V. De, and Robert L. Mercer. A tree-based statistical language model for natural language speech recognition. volume 37, pages 1001–1008, Jul 1989.

[2] Vladimir A. Bochko and Maria Petrou. Recognition of structural parts of buildings using support vector machines. In *Pattern Recognition and Information Processing, PRIP2007*, 2007.

[3] Leo Breiman, Jerome Friedman, R. A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[4] Wray Buntine and Wray Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.

[5] Bojan Cestnik. Estimating probabilities: A crucial task in machine learning. In *ECAI*, pages 147–149, 1990.

[6] Martin Drauschke and Wolfgang Förstner. Comparison of adaboost and adtboost for feature subset selection. In *PRIS 2008*, Barcelona, Spain, 2008.

[7] Martin Drauschke and Wolfgang Förstner. Selecting appropriate features for detecting buildings and building parts. In *21st Congress of the International Society for Photogrammetry and Remote Sensing (ISPRS)*, Beijing, China, 2008.

[8] Florent Fusier, Valery Valentin, Francois Bremond, Monique Thonnat, Mark Borg, David Thirde, and James Ferryman. Video understanding for complex activity recognition. *Machine Vision and Applications (MVA)*, 18:167–188, August 2007.

[9] Johannes Hartz and Bernd Neumann. Learning a knowledge base of ontological concepts for high-level scene interpretation. In *IEEE Proc. International Conference on Machine Learning and Applications*, Cincinnati (Ohio, USA), Dec 2007.

[10] Daniel Heesch and Maria Petrou. Markov random fields with asymmetric interactions for modelling spatial context in structured scenes. *Journal of Signal Processing Systems, to appear*, 2009.

[11] Lothar Hotz and Bernd Neumann. Scene interpretation as a configuration task. *KI*, 19(3):59–, 2005.

[12] Lothar Hotz, Bernd Neumann, and Kasim Terzić. High-level expectations for low-level image processing. In *Proceedings of the 31st Annual German Conference on Artificial Intelligence*, Kaiserslautern, September 2008.

[13] Lothar Hotz, Bernd Neumann, Kasim Terzić, and Jan Šochman. Feedback between low-level and high-level image processing. Technical Report Report FBI-HH-B-278/07, Universität Hamburg, Hamburg, 2007.

[14] Britta Hummel, Werner Thiemann, and Irina Lulcheva. Scene understanding of urban road intersections with description logic. In Anthony G. Cohn, David C. Hogg, Ralf Möller, and Bernd Neumann, editors, *Logic and Probability for Scene Interpretation*, number 08091 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2008. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[15] Thorsten Joachims. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and

Alexander J. Smola, editors, *Advances in kernel methods: support vector learning*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[16] F. Korč and W. Förstner. eTRIMS Image Database for interpreting images of man-made scenes. Technical Report TR-IGG-P-2009-01, April 2009.

[17] Filip Korč and Wolfgang Förstner. Interpreting terrestrial images of urban scenes using discriminative random fields. In *Proc. of the 21st Congress of the International Society for Photogrammetry and Remote Sensing (ISPRS)*, 2008.

[18] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 878–885 vol. 1, 2005.

[19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[20] Michael Mohnhaupt and Bernd Neumann. Understanding object motion: recognition, learning and spatiotemporal reasoning. pages 65–91, 1993.

[21] Bernd Neumann. Bayesian compositional hierarchies - a probabilistic structure for scene interpretation. Technical Report FBI-HH-B-282/08, Universität Hamburg, Department Informatik, Arbeitsbereich Kognitive Systeme, May 2008.

[22] David Poole, Alan Mackworth, and Randy Goebel. *Computational intelligence: a logical approach.* Oxford University Press, Oxford, UK, 1997.

[23] Ulrich Steinhoff, Dusan Omercevic, Roland Perko, Bernt Schiele, and Ales Leonardis. How computer vision can help in outdoor positioning. In Bernt Schiele, Anind K. Dey, Hans Gellersen, Boris E. R. de Ruyter, Manfred Tscheligi, Reiner Wichert, Emile H. L. Aarts, and Alejandro P. Buchmann, editors, *AmI*, volume 4794 of *Lecture Notes in Computer Science*, pages 124–141. Springer, 2007.

[24] Kasim Terzić, Lothar Hotz, and Bernd Neumann. Division of work during behaviour recognition - the SCENIC approach. In *Workshop on Behaviour Modelling and Interpretation, 30th German Conference on Artificial Intelligence*, Osnabrück, Germany, September 2007.

[25] Jan Čech and Radim Šára. Language of the structural models for constrained image segmentation. Technical Report Technical Report TN-eTRIMS-CMP-03-2007, Czech Technical University, Prague, 2007.

[26] Andrew R. Webb. *Statistical Pattern Recognition, 2nd Edition.* John Wiley & Sons, October 2002.

[27] Susanne Wenzel, Martin Drauschke, and Wolfgang Förstner. Detection of repeated structures in facade images. In Eckart Michaelsen, editor, *7th Open German / Russian Workshop on Pattern Recognition and Image Understanding*, Ettlingen, August 2007. FGAN-FOM.

[28] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *In Proceedings of the Eighteenth International Conference on Machine Learning*, pages 609–616. Morgan Kaufmann, 2001.