Features, Color Spaces, and Boosting: New Insights on Semantic Classification of Remote Sensing Images

Piotr Tokarczyk, Jan Dirk Wegner, Stefan Walk, and Konrad Schindler

***Abstract*—A major yet largely unsolved problem in the semantic classification of very high resolution remote sensing images is the design and selection of appropriate features. At a ground sampling distance below half a meter, fine-grained texture details of objects emerge and lead to a large intraclass variability while generally keeping the between-class variability at a low level. Usually, the user makes an educated guess on what features seem to appropriately capture characteristic object class patterns. Here, we propose to avoid manual feature selection and let a boosting classifier choose optimal features from a vast Randomized Quasi-Exhaustive (RQE) set of feature candidates directly during training. This RQE feature set consists of a multitude of very simple features that are computed efficiently via integral images inside a sliding window. This simple but comprehensive feature candidate set enables the boosting classifier to assemble the most discriminative textures at different scale levels to classify a small number of broad urban land-cover classes. We do an extensive evaluation on several data sets and compare performance against multiple feature extraction baselines in different color spaces. In addition, we verify experimentally if we gain any classification accuracy if moving from boosting stumps to trees. Cross-validation minimizes the possible bias caused by specific training/testing setups. It turns out that boosting in combination with the proposed RQE feature set outperforms all baseline features while still remaining computationally efficient. Particularly boosting trees (instead of stumps) captures class patterns so well that results suggest to completely leave feature selection to the classifier.**

***Index Terms*—Classification, land cover, feature extraction, pattern recognition**

I. INTRODUCTION

TODAY, the majority of humans live in cities, and it is anticipated that, by the year 2030, this will further increase to a level of 60% [1]. Developing appropriate infrastructure capable of meeting the needs of a growing population calls for constant monitoring and map updating. The high amount of small (e.g., trees) or narrow objects (e.g., footpaths) can only be captured with very high resolution (VHR) remote sensing images. Huge amounts of data are acquired with either spaceborne or airborne sensors to serve as basis for cartographic data usually being drawn manually. Manual semantic annotation is extremely time-consuming and costly, particularly if dealing with VHR images that have a high level of detail. Thus, map databases are often outdated. Moreover, in case of crisis situations evoked by, for example, earthquakes or inundations, rapid mapping is essential to organize instant response actions. In order to reduce the time and cost of mapping, automation is required - which is why one of the main goals of remote sensing research in the past decades has been semantic classification. Despite large efforts over the last 30 years, this task remains largely unsolved.

What makes semantic classification challenging particularly in urban areas is the presence of many object classes within a single VHR image, the often small extent and heterogeneous structure due to fine-grained texture details, and the strongly varying between-class dependences. This leads to a large within-class variability while keeping the between-class variability at a low level at the same time. Here, we view semantic classification in a supervised setting that assigns one class label to each pixel in the image. The different class labels correspond to the specific object categories needed for mapping which depend on the application at hand. In this paper we group urban objects into four categories: buildings (of arbitrary size), streets (including bare ground), high vegetation, and low vegetation.

In general, a supervised classification consists of four main steps: 1) feature extraction from the raw input data; 2) classifier training on a subset of the input data with labeled ground truth; 3) validation on another subset with ground truth; and 4) classification of the data using the previously trained and validated classifier. Although a vast body of literature deals with the last two steps considering mostly machine learning methods like support vector machines (SVMs) [2], fuzzy approaches [3], random forests [4], [5] or boosting [6], less systematic attention has been paid to the design and selection strategies of features. Usually, raw pixel intensities of all spectral bands are used, often enriched with features returned by the Normalized Differential Vegetation Index (NDVI) or some texture filter bank. Because it is, in general, not known in advance which combinations or mapping functions of features would help separating object classes optimally, manual feature selection can be viewed as ”educated guessing,” although certain features are, of course, physically motivated (e.g., NDVI).

We propose to couple feature selection and training in such a way that we allow a boosting classifier to directly choose those features, that best discriminate the target classes, from an extensive set of feature candidates. Instead of learning a highly complex feature set like done in deep learning [7], [8] or with texture priors in the framework of Markov random fields [9], [10], we aim at providing a quasi-exhaustive feature

The authors are with the Photogrammetry and Remote Sensing group, Institute of Geodesy and Photogrammetry, Swiss Federal Institute of Technology, 8093 Zurich, Switzerland, e-mail: (piotr.tokarczyk, jan.wegner, stefan.walk, schindler)@geod.baug.ethz.ch

bank that is capable of adapting to scene specific properties in terms of radiometry, object classes etc.

Since the seminal work of Viola and Jones [11] (and similarly that of Simard *et al.* [12]), integral images have become very popular to compute box filters to approximate smooth filters. For example, Bay *et al.* [13] make use of this concept to approximate Hessians in order to speed up computation of Speeded Up Robust Features (SURF) descriptors. We make use of integral images for averaging and approximating derivatives in order to efficiently extract this feature set despite its high dimensionality. Boosting has the advantage that the extraction of the whole feature set is only done once for a rather small training area whereas for testing, only the most discriminative subset of features explicitly selected during training has to be extracted. More precisely, features that do contribute only very little are completely discarded as opposed to down-weighting like in case of SVMs for example. Computational speed, memory requirements, and classification accuracy further benefit from computing the majority of features randomly regarding patch sizes and location. These so-called randomized quasi-exhaustive (RQE) features serve as input to boosting.

In order to evaluate the proposed feature/classifier setup we perform an extensive set of experiments on five VHR urban scene images with three spectral bands. Cross-validation ensures that any bias caused by a specific training/testing setup is minimized. The goal of this study is threefold: 1) to compare the proposed RQE feature bank to various baseline features; 2) to evaluate the influence of three different color spaces; and 3) to test boosting decision stumps (linear) versus boosting decision trees, which, on the contrary to stumps, can represent nonlinear decision boundaries.

II. Related work

Although feature design and selection has recently led to significant progress in computer vision and machine learning (e.g., [13–17]), the classification of VHR remote sensing data is usually still done based on standard features like raw pixel intensities. In [18], it is advocated to include intensities within a certain neighborhood around the pixel of interest for larger support. Other studies that use VHR aerial imagery to classify multiple object classes in urban scenes (e.g., [19–22]), additionally exploit height information obtained either from airborne-laser-scanning platforms or dense image matching.

One strategy for discriminating different object classes is to model texture via prior distributions of graphical models [9], [10]. Helmholz *et al.* [23] apply the Markov random field of [9] to characterize texture for the classification of aerial and satellite images.

A more common approach to texture description is the banks of texture filters (e.g., [24] and [25]). The image channels are convolved with filters (e.g., approximations of Gaussians, Laplacians, etc.) and their responses act as features during classification. A large variety of texture filter banks exists and has been applied to various tasks, for example, image segmentation [26], [27] and image registration [28]. Although being commonly used for pattern recognition problems in computer vision, filter banks have been less frequently adapted to remote sensing data. One of the few examples is the work in [29] where the performance of Gabor filters on aerial images was evaluated.

Generally, banks of filters are rather straightforward, and they quickly lead to relatively promising results. However, performance varies depending on the type of texture [30] because each filter bank is tuned for a specific application (i.e., a certain kind of texture). Thus, there is a need for a different approach that is capable of automatically adapting to various kinds of texture.

One concept that aims to resolve this problem is deep learning where (nonlinear) feature extractors are learned directly (e.g., [7], [8], and [31]). This has been applied in [32] and [33] to patch-based road classification in aerial images. However, in a recent evaluation [34] exploiting deep learning for feature extraction to classify VHR remote sensing data (with random forests), we found that deep learning is brittle to set up, and often, results are not improved compared to simple linear filter banks. In this paper, we thus resort to an alternative (linear) and somewhat more intuitive strategy by providing a quasi-exhaustive set of candidate features to the classifier instead of learning these directly from raw pixels.

Alternatively, approaches exist that first compute a very large comprehensive set of features and, second, reduce the feature space dimensionality before training by partial least squares [35], [36] or genetic algorithms [37], [38] for example. In [39] Dollár *et al.* propose to meaningfully organize the potentially huge comprehensive space of features in such a way that the most informative features can be picked by a boosting classifier without having to search the entire feature space brute force. Experiments on human face and pedestrian benchmark data indicate superior performance compared to manually chosen feature sets. As opposed to selecting a discriminative subset of features prior to training, Dollár *et al.* [40] propose to let the classifier select discriminative features directly during training. Using integral images, they randomly generate a vast number of box filters and record their responses within a detection window. AdaBoost then selects the most discriminative features, thus largely improving performance over standard baseline features. Note that the authors also report better performance compared to separate feature mining and learning as proposed earlier in [39]. A similar approach, but using random forests instead of boosting, was proposed by Fröhlich *et al.* [41] and applied to satellite images in [42]. The authors randomly sample Haar-like features inside a window, which serve as input to an augmented random forest classifier. Contextual class-specific cues are learned iteratively based on previous classifier outputs. Another good example of joint optimization of classification and feature selection in the spectral dimension is shown in [43]. The authors employ the generalized relevance learning vector quantization in order to distinguish between classes in the hyperspectral remotely sensed imagery.

A major advantage of doing feature selection and training jointly is that the classifier directly selects those features that it finds to be most discriminative, i.e. those that work best in combination with this particular classifier, data set, and object classes. Our approach is inspired by [40] and follows

similar lines of thought (in terms of the comprehensive feature candidates set) as [41] and [42].

III. METHODS

It is common practice in remote sensing to treat feature selection and classification as separate subsequent problems. In this paper, conforming to recent achievements in the field of machine learning, we pose these problems jointly. In contrast to the standard stepwise procedure, appropriate features for characterizing a particular scene are *not* being selected manually. This step is entirely left to the classifier that selects only those features from a large candidate set of RQE features, which minimize the misclassification error for a specific dataset.

In the following subsections, we describe the features which serve as baselines for our experiments. Thereafter, we turn to a detailed description of the proposed RQE features that serve as candidate set for the boosting classifier, which is explained in the last part of this section. Emphasis is put on the differences between boosting stumps and boosting trees.

A. Baseline features

Here, we describe the standard features used in remote sensing for aerial- and satellite-image classification. Usually, the user makes an *"educated guess"* about which specific features might be the best to classify a particular scene. While this procedure is straightforward, discriminative features not anticipated by the expert will not be computed, and class-specific patterns will possibly remain hidden. We propose to avoid *ad hoc* manual feature selection by letting the classifier choose features directly during training. In the following we describe standard features that will act as baselines for experiments.

1) Raw Pixel Intensities: The most basic feature of a digital image is the intensity of a single pixel. Today's digital airborne and spaceborne optical sensors capture sun radiation in multiple spectral bands. Each band is recorded in a single channel, leading to several intensities per pixel, one for each channel. Standard VHR multispectral aerial sensors have four spectral channels (VHR multispectral satellite sensors usually have more): red, green, blue and infrared. If we directly take the intensities of all available channels per pixel, the feature space dimension is equivalent to the number of channels (e.g., four in case of standard sensors). The advantage of these features is that they are intuitive and very fast to compute.

2) Raw Pixel Intensities Within 15×15 Neighborhood: The natural extension of taking single intensities over all channels per pixel is to include a certain neighborhood. State-of-the-art VHR data has a ground sampling distance (GSD) between 0.05 and 0.5 m. Although such high spatial resolution is of greatest value for urban remote sensing applications, it creates new challenges because a lot of small objects emerge that remain hidden in images of lower resolution. Single trees are composed of tens of pixels, and small superstructures on roofs (chimneys, dormers, etc.) become visible. Cars on the streets no longer consist of one or very few pixels but are clearly recognizable. As a consequence, the spectral variability within object classes considerably increases, whereas the interclass variability remains low. A common strategy to resolve potential ambiguities is discriminating classes based on their specific textural patterns. The most natural and straightforward approach for texture learning is to add intensities of neighboring pixels to such of the pixel of interest. More precisely, intensities of adjacent pixels within a square window are simply added to the feature vector. Usually, the window size varies between 3×3 to 21×21 pixels and is closely linked to image resolution and object size. We have tested a wide range of window sizes and found 15×15 to be sufficient for our data GSD, creating a 225-dimensional feature vector per channel. For input images with three channels, we thus obtain a 675-dimensional feature space. Note that, due to the strong overlap of neighboring windows, feature vectors of adjacent pixels are highly correlated. In turn, classification results can be expected to be a lot smoother than such based merely on single pixels.

3) 15×15 Pixel Neighborhood + NDVI: Along with the neighborhood of each single pixel, as described in the previous subsection, we add an NDVI channel. The NDVI is a standard feature created by nonlinear combination of the red and near infrared channels and is widely used in optical remote sensing. It captures differences between vegetated and nonvegetated areas. In our experiments, we consider the NDVI as independent channel, i.e. all NDVI values in a 15×15 neighborhood of the pixel of interest are recorded. Adding the 225-dimensional NDVI feature vector to the already existing one of the three spectral bands thus results in a 900-dimensional feature space.

4) Texture Filters: Texture filter banks are another standard approach to capture object class patterns. They are a good example for "hand-crafted" features where an expert designs a relatively small bank of specifically engineered features to capture expected object class patterns. Images are convolved with a series of filters and their responses are stored in a feature vector. A good representative for this kind of technique is the subset of the root filter set (RFS) used in [44], and we take it as baseline for our experiments.

All channels are convolved with the filters adopted from [44]: four Laplacian-of-Gaussians (LoG), three Gaussians and four first-order Gaussian derivatives. Four LoG and three Gaussian kernels are computed in different scales ($\{\sigma, 2\sigma, 4\sigma, 8\sigma\}$ and $\{\sigma, 2\sigma, 4\sigma\}$). As the first-order Gaussian derivatives are computed independently in two different scales $\{2\sigma, 4\sigma\}$ and two directions (x and y), they produce four responses. In total, 11 features per channel are computed, generating a 33-dimensional feature space when using input images with three spectral bands. The σ value has been empirically tested, and we found $\sigma = 0.7$ to yield best results. Note that, by convolving all channels with the described filters, a neighborhood of each pixel is implicitly considered.

B. RQE Features

Our aim is to avoid data-specific design of features completely. Therefore, we generate a vast and redundant set of simple features, that can be computed efficiently via integral images [11], to allow the classifier to choose the most discriminative ones directly. This comprehensive RQE feature

bank enables one to handle different lighting conditions, object class properties, different scene structures, etc., thus mitigating scene-specific performance variations of standard filter banks [30]. This approach can be viewed as an intuitive intermediate step between deep learning methods [7], [8], and [31–33]) and standard feature banks (e.g., [24], [25], and [44]).

A sliding window that is shifted pixel-wise across the image acts as basic unit. All following operations take place inside this window and are exactly the same at each of its positions. Various window sizes were tested, and 15 × 15 performed best across all data sets.

The first subgroup of the RQE feature set consists of differences between pairs of patches. Instead of allowing only predefined regular patterns and patch shapes (as in [45]), we generate a large pool of features by randomly choosing patch size and position (inspired by [40]) within the 15 × 15 window. Each randomly chosen patch is being mirrored with respect to the central pixel of the window, and the difference between the mean intensities of both patches is calculated. Patches can take on any rectangular shape; sizes can vary from 1 × 1 to 15(14) × 14(15) (see Fig. 1). Randomizing patch generation allows us to capture a wider range of textures while reducing runtime and memory, too. Note that this randomized patch sampling is done only once. The configuration then remains fixed for all positions of the sliding window, i.e. patch sampling is *not* repeated for each new window position.

Note that this definition of patch generation explicitly allows overlapping patches, both in the same iteration step and in consecutive steps. For example, consider a patch that has been randomly generated such that it overlaps with the central pixel of the 15 × 15 window. If we now mirror this patch at the central pixel, the second patch will overlap with the first one. In the following iteration, the newly generated initial patch may then overlap with either the first or the second patch of the previous iteration, etc. While this feature generation strategy results in highly correlated features, it is, in principle, capable of covering isotropic AND anisotropic texture patterns. Designing symmetric filters accounts for the nadir perspective of remote sensing data. As opposed to terrestrial images, where a typical order of object classes naturally exists (e.g., sky appears in the top image part, a building facade in the middle, and cars at the bottom), the same scene acquired in nadir view does not have such a simple ordering. One could argue that still some typical organization of object classes exists (e.g., car on road and buildings close to streets), but we cannot assume any scene-specific structure in terms of ”above,” ”below,” ”left,” and ”right”. Recall that flight paths of aerial sensors vary and urban structures do often not follow any preferred direction (except cities with regular road grids). Thus, assymmetric filters would largely increase feature space dimensions, resulting in longer training times without significant information gain.

In addition to these differences based on randomly generated patch pairs, the second subgroup of the RQE feature set contains the following.[1]

- All pixel intensities within the 15 × 15 window in three different scales: 1) raw intensities per pixel (675 features); 2) the same window after 3 × 3 mean-filtering (675 features); and 3) after 5 × 5 mean-filtering (675 features). We also add mean values for 3 × 3 and 5 × 5 up to 15 × 15 patches, but only for the central pixel of the 15 × 15 window (21 features). Note that this is different from filtering the whole window (as in case of 3 × 3 and 5 × 5 mean-filtering), which would largely increase the feature space dimensionality without significant performance gain.
- Differences between mean values of square patches of the same size centered on the central pixel of the 15 × 15 window between different spectral channels [see Fig. 2 (right)]. Square patch sizes range from 3 × 3 to 15 × 15 (42 features).
- Differences between mean values of square patches of different sizes (ranging from 3 × 3 to 15 × 15), but only within the same spectral channel (126 features) [see Fig. 2 (left)].

The first and the second subgroup contain only linear features. To enable a fair comparison with the NDVI, which is a nonlinear combination of intensities across channels, we add a third nonlinear subgroup. This will also allow us to experimentally evaluate the general impact of nonlinear features, which do, of course, increase the feature dimensionality, on the classification performance. We include and extend the NDVI by computing it for all possible combinations of two channels. More precisely, for two different channels, we divide the difference of mean values of square patches by their sum. These square patches (ranging from 3 × 3 to 15 × 15) are centered on the central pixel of the 15 × 15 window, and both patches of a pair (i.e., one in the first channel and another one in the second channel of the channel combination) have the same size (42 features).

The second subgroup of the RQE feature set with only linear features consists of 2214 features. Adding the third subgroup, the nonlinear features, increases the feature space to 2256 dimensions. Recall that patches of the first sub-group are randomly sampled within the 15 × 15 window, and thus, feature space dimension depends on the number of samples.

This RQE feature set has inherited all the properties of [45], but it broadens them. True derivative filters are approximated reasonably, and the randomized patch-based subgroup naturally enables including a vast range of scales and texture frequencies. Patches are no longer restricted to a grid, and their shape is not imposed to be square. This new layout also takes advantage of color information explicitly by exploiting differences (and ratios) between spectral bands.

IV. Classifier

The classifier that is chosen is a multiclass extension of discrete AdaBoost [46]. Boosting has turned out to be significantly better (4%–10%) than linear SVMs and random

[1]The numbers of features in brackets are always given under the assumption of three channels.

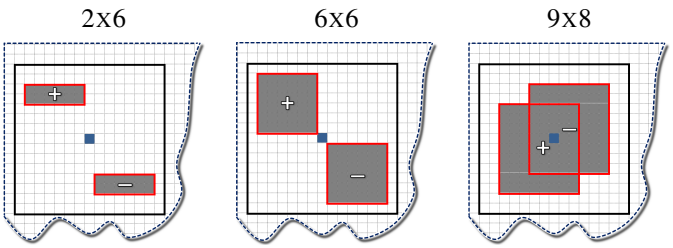

Fig. 1. Randomized patches: Examples of rectangular patches of random size. The grid within dashed lines symbolizes the image, solid black lines depict a sliding window of size 15×15 centered on the pixel of interest (dark blue). The difference is computed between the mean values ("-" subtracted from "+") of both image patches (red) and repeated with random position and size of patches.

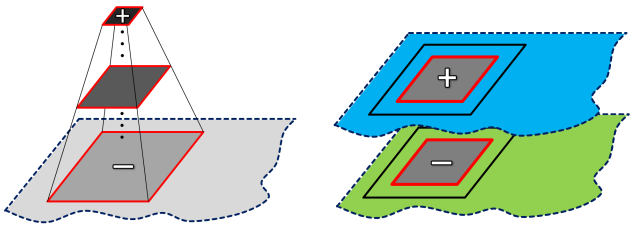

Fig. 2. Left image: Differences between means inside square patches with different sizes (3×3 (dark grey) to 15×15 (light grey)) centered on the pixel of interest. Right image: Differences between means inside square patches (red) of equal size over different image channels (within the sliding 15×15 window (solid black line)).

forests in previous experiments for us. Kernelized SVMs have prohibitive memory and runtime requirements, so they have not been considered. AdaBoost combines multiple classifiers (so-called "weak learners") in an iterative fashion. If the weak classifiers (like decision trees) operate on a subset of features during testing, AdaBoost implicitly performs feature selection during training. In each iteration, the training data are reweighted so that samples that are hard for the current classifier obtain a larger weight. Then the weak learner that reduces the weighted error rate the most is added to the classifier, and the next iteration starts. In image processing, weak learners for AdaBoost are usually decision trees, with decision stumps (trees with only one split and two leaves) being the most common choice.

A. AdaBoost

AdaBoost works by combining K weak classifiers h_k into a strong classifier H. The final classifier (with the feature vector $\mathbf{x}$) is a linear combination of the weak classifiers

$$H(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k h_k(\mathbf{x}) \tag{1}$$

The classification result (if the sample is predicted in the positive or negative class) is the sign of H. The magnitude of H signifies the confidence of the classification result.

AdaBoost starts with equal weights ($\frac{1}{N}$) for all N samples. In each iteration AdaBoost tries to find the best weak learner to add to the strong classifier by looking at the weighted error rate ϵ. ϵ is the sum of all weights for which the weak learner h disagrees with the class label y_i. The weak learner that maximizes $|\epsilon - 0.5|$ is chosen for the next iteration and is assigned a weight α

$$\alpha = \frac{1}{2} \log \left(\frac{1-\epsilon}{\epsilon} \right) \tag{2}$$

The better the weak learner, the higher the weight it gets in the final strong classifier. After the weak learner is added, the weights for each training sample is multiplied by a factor of $\exp\left(-\alpha y_i h(\mathbf{x_i})\right)$, which is bigger than one if the weak learner disagrees with the ground truth label ($y_i \neq h(\mathbf{x_i})$) and smaller than one otherwise. This is the mechanism that ensures that "hard" samples are more likely to be classified correctly by the weak learner chosen in future iterations.

Before the next iterations, the weights are normalized to sum to one. Then, the weak learner that works best for the updated weights is added, and the whole procedure is repeated until no weak classifiers that operate better than chance are found or the desired number of weak classifier is reached.

As standard AdaBoost is a binary classifier, discriminating a "positive" (label $y = +1$) from a "negative" (label $y = -1$) class, and we are working with a multiclass scenario, we need to use a modification of it that allows more than two classes. In this paper, we use a variant of AdaBoost.MH [47], which is a multiclass extension to AdaBoost.[2] By using a one-versus-all strategy (as known, for example, from multiclass SVMs), it is possible to reduce a C-class problem to a set of C binary problems. Instead of being scalar-valued, $\mathbf{H}$ and $\mathbf{h}$ are now vector-valued functions with C components. The labels also become vectors. The label y_i of a sample with class c has $+1$ in its cth component and -1 in the other components. Weak classifiers are of the form

$$\mathbf{h}(\mathbf{x}) = \mathbf{d} \cdot \eta(\mathbf{x}) \tag{3}$$

where $\mathbf{d}$ is also a vector with c components with $+1$ or -1 in each component and η is a binary classifier (as used in standard AdaBoost). The cth component of $\mathbf{d}$ is $+1$ or -1 if the correlation (under the influence of the sample weights) between the binary classifier η and the true class label is positive or negative, respectively. This construction means that each weak classifier tries to separate two sets of classes (the classes with $+1$ in $\mathbf{d}$ from those with -1), so we can expect that we need more weak classifiers than in the two-class case (by a factor of $\log_2 C$) just by construction (since each weak learner is still effectively binary).

The weights are also vectors whose components get updated independently of each other in the same fashion as in (2), so the weight increases for the components where $\mathbf{h}$ disagrees with the class label and decreases otherwise. There is also a per-class weighted error rate, and the "best" weak learner is the one that has the lowest weighted error rate averaged over all components. The predicted class in $\mathbf{H}(\mathbf{x})$ is the one whose component has the highest value. A possible confidence measure is the distance between this value and the second highest value.

If AdaBoost.MH were applied to a binary classification

[2]In the experiments, the software of [48] was used.

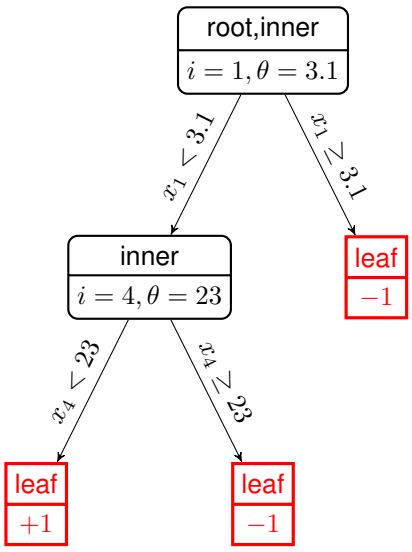

Fig. 3. Example decision tree with three leaves

problem, the two components of each vector-valued variable would be equal in magnitude with different signs. The component of the "positive" class would be equal to a standard AdaBoost classifier, and the component of the "negative" class would be its inverse.

As mentioned, weak learners used for AdaBoost are usually decision trees, often decision stumps. In general decision trees are able to handle multiclass problems; however the trees used in AdaBoost and AdaBoost.MH produce a binary decision – either -1 or $+1$.

B. Decision Trees

We will describe stumps and trees with standard AdaBoost terminology for notational simplicity, as for AdaBoost.MH averages over vector components are necessary in multiple places. A decision tree consists of inner nodes (splits) and end nodes (leaves). A decision is reached by starting at the root and applying the rules in the inner nodes until a leaf is reached. ℓ is the decision function for nodes. Nodes defer to their child nodes if they are inner nodes. Which child node is chosen is determined by the splitting rule. The splitting rules are in our case simple thresholds on single feature dimensions. The decision of a leaf is $+1$ if the sum of weights of positive samples μ_+ that ended up in this leaf during training is bigger than the sum of weights of negative samples μ_-, and -1 otherwise.

$$h(\mathbf{x}) = \ell_{\text{root}}(\mathbf{x}) \tag{4}$$

$$\ell_{\text{inner}}(\mathbf{x}) = \begin{cases} \ell_{\text{left}}(\mathbf{x}) & \text{if } x_i < \theta \\ \ell_{\text{right}}(\mathbf{x}) & \text{otherwise} \end{cases} \tag{5}$$

$$\ell_{\text{leaf}}(\mathbf{x}) = \begin{cases} +1 & \text{if } \mu_+ > \mu_- \\ -1 & \text{otherwise} \end{cases} \tag{6}$$

Each inner node has its own parameter set of i (the feature index) and θ (the threshold). An example tree is shown in Fig. 3. During training, trees are grown top-down. The tree learner seeks to find a combination of a feature index i and a threshold θ that splits the set at a node in a way that decreases the weighted error rate the most. With μ_+ and μ_- being the sum of weights of positive and negative samples in a leaf node, the weighted error in that node is given by

$$\epsilon_p = \min(\mu_+, \mu_-). \tag{7}$$

This is because the label the node predicts corresponds to the label with the bigger sum of weights, so the smaller sum corresponds to the weighted error.

A beneficial split has a $+1$ label on one side and a -1 label on the other side, as a split with the same label for both child nodes would not decrease the error rate. We call the sums for the weights in the left and right child nodes of a potential split $\mu_{l,\cdot}$ and $\mu_{l,\cdot}$, respectively. This leads to the error after splitting as

$$\epsilon_s = \min(\mu_{l,+} + \mu_{r,-}, \mu_{l,-} + \mu_{r,+}) \tag{8}$$

where the first part corresponds to a label of -1 for the left label and 1 for the right label, and the second part for the other case. The gain by splitting is

$$\gamma = \epsilon_p - \epsilon_s. \tag{9}$$

To train the tree, we start with a root node that contains each training sample (the number of leaves is 1 at the start). At each iteration, we examine for each leaf node the possible gain that can be achieved by splitting. We split the node with the highest gain and repeat the procedure until we get the desired number of leaves. In the case of stumps, we only perform one split, meaning that we only have two leaf nodes. For decision trees, we perform three splits leading to four-leaf nodes. Since the decisions along the path perform an AND-combination of the split rules, this enables the four-leaf decision trees to capture correlations between features, something that decision stumps can not capture (because the final classifier is a linear combination of simple thresholds over single features).

V. EXPERIMENTS

We perform extensive experiments on five different data sets to do the following: 1) compare the different feature sets; 2) to evaluate the impact of different color spaces in this context; and 3) to assess whether we gain any performance if boosting trees instead of stumps.

We select four standard categories that incorporate many subcategories: buildings, high vegetation (trees, high bushes, etc.), low vegetation (grass, low bushes, etc.) and streets (including cars). The ground truth for all datasets used in this evaluation was labelled manually.

In order to minimize the bias of specific train/test splits, we perform fivefold cross-validation for each data set as recommended in machine learning and pattern recognition literature (e.g., [49], [50]). All five datasets are divided into five vertical strips of equal size. Subsequently, the classifier is trained on features extracted from four different strips and tested on the fifth. This procedure is repeated five times, each time with another training/testing configuration. Finally, quality numbers for all five runs are averaged.

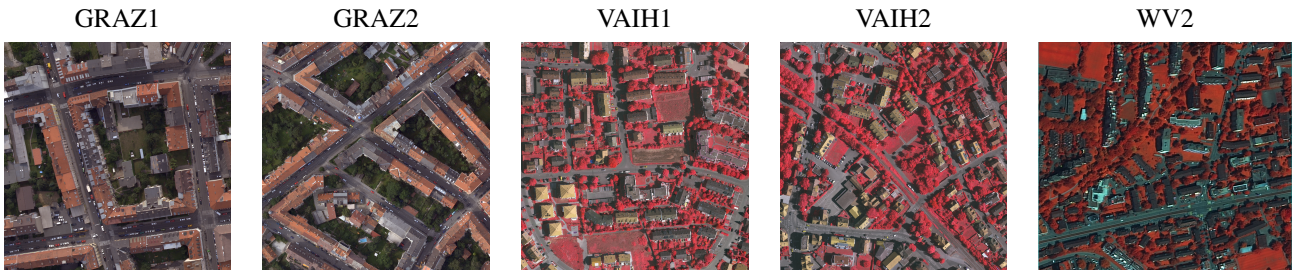

Fig. 4. Original images used for the experiments.

To speed up feature extraction and training, we do not extract features for all pixels in the training strips, but select pixels randomly across the four training strips. Note that all pixels within a 15 × 15 window (i.e., the size of the sliding window used for feature extraction) are highly correlated anyway. Therefore, we do not expect to loose any significant information if using a subset.[3]

For training the boosting classifier, we use single stumps or decision trees as weak learners. Three parameters of the classifier had to be chosen manually: the number of boosting iterations, the number of extracted random features per channel, and the number of leaves in the decision trees. We tested numerous choices of the latter two and found 500 extracted random features per channel and four leaves per decision tree to be good compromises between classification accuracy and computation time. The maximum number of boosting rounds was set to 500 iterations, but it should be noted that, in practice, most training runs converge much earlier.

A. Data Sets

We perform the evaluation of the proposed method on four aerial images and one satellite image (see Fig. 4). Test images GRAZ1 and GRAZ2 have dimensions 800 × 800 and 1000 × 1000 pixels, respectively. Both are subsets of a large RGB image block acquired with a Microsoft Vexcel Ultracam D camera of the city of Graz (Austria) at a GSD of 25 cm. These images show a typical European city center with narrow streets, dense building formations and vegetation. Due to the absence of a near infrared band, the pseudo-NDVI was computed for the tests that would use NDVI, replacing near infrared with the green channel.

Both images VAIH1 and VAIH2 are 1000 × 1000 pixels wide. They are a subset of a true-orthophoto mosaic (with GSD = 20 cm) acquired over the town Vaihingen an der Enz (Germany) using an Intergraph DMC sensor with red, green, and near infrared channels. Typical suburban areas of a European city are mapped with a majority of detached single-family houses and lots of vegetation. This dataset is publicly available as part of a benchmark data set for urban object classification and 3-D building reconstruction [51], [52].

Our satellite image (WV2) is a 1000×1000 pixels subset of a WorldView-2 stereo scene over the city of Zurich (Switzerland), showing the suburban parts of the city with mainly high buildings (blocks of flats). The image was pan-sharpened and thus has a GSD of 50 cm. For the orthophoto generation, we choose three out of eight spectral channels: red, green, and near infrared.

B. Color Spaces

In order to evaluate whether a specific choice of color space does have an impact on classification results, we compare the performance of the classifier with features extracted in three different color spaces: 1) standard RGB[4]; 2) data set-dependent, principal component analysis (PCA)-derived channels which can be viewed as a relative color transform because it adapts to each training image and decorrelates channels; and 3) YUV, a linear transform of RGB that separates intensity from color information (also known as YCbCr).

The YUV color space is primarily used in video applications. This model defines a color space as consisting of one luminance (Y) and two chrominance (UV) components. The two latter ones are the differences between the blue (U) and red (V) components of the original image and a reference value [53].

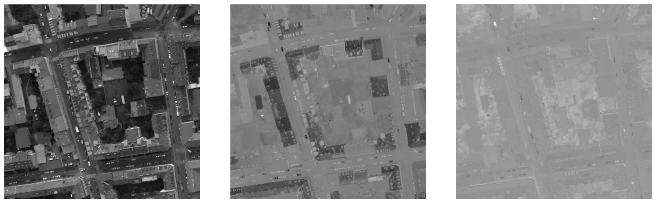

Fig. 5. PCA example with (from left to right) the first, second, and third PCA components of the GRAZ1 image shown left in Fig. 4.

The PCA is a method commonly used in remote sensing for satellite image destriping, data decorrelation, or reduction of the data dimensionality [54]. Here, as our goal is none of the above, we take the advantage of its ability to perform the orthogonal transformation of the image's original color space into a linearly uncorrelated one. The first channel of the new color space will have the largest possible variance, and each subsequential channel has the highest variance possible under the constraint that it is again orthogonal to the preceding channels. It has to be emphasized that, because of the *data-driven* nature of this method, the parameters of the transformation

[3]Note that one fifth of all training pixels corresponds to 200k pixels in case of 1000×1000 images and to 128k pixels for 800×800 images, which was found to deliver sufficiently stable features for training.

[4]We use the label "RGB" also for the datasets where the image channels are red, green, and near infrared. The available channels for each dataset are described in the previous section.

are dependent on the input data and potentially adapt better to lighting and scene. An example for the GRAZ1 image shown in Fig. 4 is given in Fig 5. It can be seen that the first PCA channel mainly contains intensity information of the image and the second one its degree of redness.

C. Results and Discussion

We show direct per-pixel overall classification accuracy results of the boosting classifier achieved with either decision stumps or decision trees. Results after cross-validation per color space are shown in Table I. Per-class user's and producer's accuracies are presented in Table IV. The CPU times used for feature extraction, training decision stumps, training decision trees, and testing the classifier are shown in Table II, respectively. One cutout of each of the five original images (see Fig. 4) is shown in Fig. 10 together with the corresponding manually labeled ground truth. For these five cutouts, we display classification results (boosting decision trees) for RGB in Fig. 8 and for PCA in Fig. 9. Note that we neither did any pre-processing of the input images (e.g., segmentation into superpixels and noise filtering) nor any postprocessing after classification (e.g., morphological cleaning and smoothing prior) to avoid biasing the results. To test for the statistical significance of the results, we employ the Wilcoxon signed-rank test [55]. It was chosen over the paired Student's t-test because it does not make the assumption that the differences in classifier performance are normally distributed. We test five datasets with five cross-validation runs each, resulting in 25 data points for the comparison of two different classifiers. When comparing families of classifiers (i.e. stumps against trees), more data points are available. We use the 99% confidence interval ($p < 0.01$) to decide if two classifiers are significantly different. All significance tests are for boosting trees with 500 iterations unless otherwise specified. Boosting decision trees instead of stumps yields significantly ($p \approx 10^{-70}$) better results (see Table I). We plot the overall classification accuracy versus boosting iterations for all color spaces and separate stumps from trees (see Fig. 7). For all three color spaces, it becomes apparent that boosting four-leaf trees outperforms stumps in terms of the final achievable accuracy after 500 iterations (see zoomed windows inside the graphs). It seems that trees somewhat better capture discriminative patterns in features per object class, i.e. feature structures and dependencies can be better expressed with trees. One possible explanation is that trees with more than one level are capable of modeling more complex class boundaries in feature space directly. Another interesting observation underpining this clue is that boosting decision trees reduces the accuracy gap between the different input feature sets. In general, simpler features (e.g., 15×15 pixel neighborhood) gain more accuracy by using decision trees. On the downside, boosting decision trees takes longer.

The choice of color space does only have a small impact on classification performance. Whether or not the differences are significant depends on the choice of feature. Classifiers using a 15×15 neighborhood in YUV and PCA obtain significantly ($p \leq 0.002$) better results over using RGB, but when NDVI is added to RGB, the difference becomes insignificant (cf. Fig. 7). When using our RQE features, using PCA results in a slight (0.5%–0.7%) but significant ($p \leq 0.002$) improvement over both RGB and YUV. There is no significant difference between RGB and YUV when used with the RQE features. These effects can be seen, for example, if we visually compare the classification results of the subscene of GRAZ2 for RGB (see Fig. 8) and PCA (see Fig. 9). Boosting with RQE or 15×15 features better separates building roofs and street, leading to generally smoother results.

The proposed RQE feature bank either outperforms the baselines or performs identically. In Fig. 6, we accumulate ranks one to seven for RGB[5] and one to five for PCA and YUV, i.e. we count how often a feature set ranks first, second, third, etc. based on the results in Table I. Note that we only provide rankings for tree boosting per color space, where the performance gap between different feature sets is smaller than for stump boosting. These cumulative diagrams underline that the proposed RQE features most often rank first or second. However, the real performance gap in terms of percent points is small. Often, differences between the top three methods are below one percent point (e.g., see Table I). In terms of statistical significance, including the nonlinear RQE features does not result in a significant improvement. For the RGB colorspace, we observe that RAW $\approx$ RAW+NDVI $<$ WINN $< 15 \times 15 <$ 15×15+NDVI $\approx$ RQE linear $\approx$ RQE nonlinear. In the other color spaces, we get similar results (without the NDVI features). When using PCA, the 5% improvement that WINN features provide over using raw pixels is interestingly actually not statistically significant ($p = 0.15$). While WINN is slightly better on some crossvalidation folds, RAW is over 5% better than WINN on a few other folds, leading to this result. Also, when using PCA or YUV, 15×15 features are not significantly worse than RQE features.

In most cases, boosting with RQE features achieves a given accuracy sooner than other methods (see red and orange curves in graphs in Fig. 7). It turns out that this faster convergence of boosting with the RQE feature bank is mainly due to the averaging patches like displayed in Fig. 1. A closer look at the features being chosen by the boosting classifier (see Table III), reveals that, in the later iterations, the classifier focuses on smaller details and choses features composed of single-pixel values. This suggests that coarser features help the boosting classifier capturing lower scale structures first before extracting fine-grained details. Particularly in conjunction with boosting, which ranks features with respect to their ability to distinguish between the object classes, this strategy is beneficial.

The subset of the RFS as used in [44] (WINN), which is a good representative of standard texture filter banks, does perform better than single raw pixel values (compare blue curves with brown/green curves in Fig. 7) but worse than all other features. Interestingly, these filters do not only perform inferior compared to the RQE filter bank, which can be expected because RQE represents a much wider range of textures, but also compared to raw pixel intensities within a

[5]Recall that we tested two more input feature sets for the RGB color space (compared to PCA and YUV) because we added the NDVI to raw pixel intensities and to such of a 15×15 neighborhood.

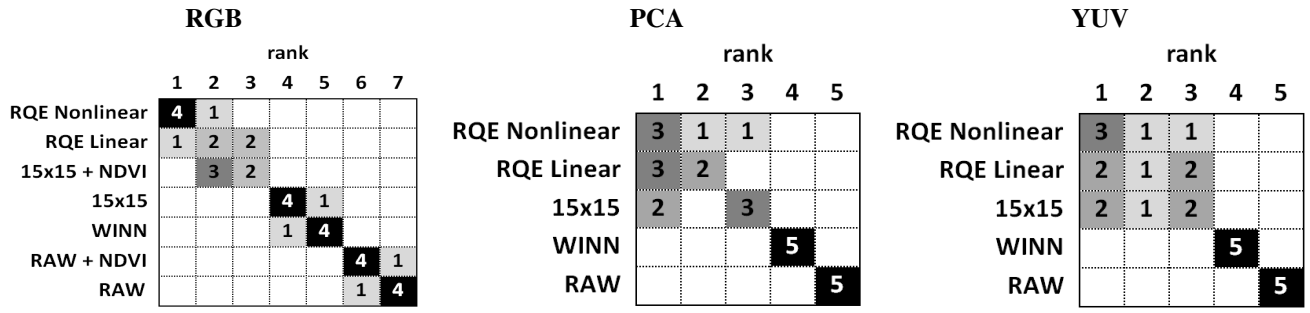

Fig. 6. Cumulative ranking of the feature sets per color space (for trees). Darker colors indicate higher scores at the corresponding rank (1-7 for RGB, 1-5 for PCA and YUV), exact numbers are written inside cells.

RGB	GRAZ1		GRAZ2		VAIH1		VAIH2		WV2	
	DS	DT	DS	DT	DS	DT	DS	DT	DS	DT
RAW	66.9	69.7	71.3	73.7	72.7	74.6	66.2	68.1	68.2	69.6
RAW + NDVI	69.8	71.6	72.5	74.2	73.0	74.6	66.9	68.2	68.2	69.4
15×15	71.3	74.0	77.1	81.4	78.9	81.5	73.8	78.1	75.4	78.4
15×15 + NDVI	74.4	77.2	78.6	**82.6**	79.3	**82.0**	75.2	78.4	75.7	**78.7**
WINN	70.2	74.1	75.7	79.0	76.5	79.2	71.3	74.8	72.9	77.9
RQE LINEAR	**74.5**	**78.2**	**79.8**	**82.6**	**80.3**	81.9	**75.9**	**79.5**	**76.5**	78.6
RQE NONLINEAR	**75.1**	**78.3**	**80.2**	**82.8**	**80.2**	**82.0**	**76.2**	**79.4**	**76.4**	**78.7**
PCA										
RAW	69.9	72.3	72.1	74.4	74.3	75.4	66.8	69.5	68.8	70.1
15×15	75.9	79.1	78.7	83.0	80.3	**82.6**	74.1	79.5	76.5	**78.8**
WINN	72.3	75.7	77.5	80.9	76.0	78.6	70.4	74.6	74.1	77.2
RQE LINEAR	**77.0**	**79.8**	**80.8**	**83.3**	**80.9**	**82.6**	**75.8**	**79.7**	**77.0**	**78.8**
RQE NONLINEAR	**77.9**	**80.1**	**80.8**	**83.1**	**80.8**	82.4	**76.1**	**79.9**	**77.5**	**78.8**
YUV										
RAW	69.1	71.9	72.1	74.1	73.8	75.5	66.0	69.4	67.6	69.7
15×15	74.2	**79.2**	77.6	82.7	79.2	**82.3**	**75.9**	**79.4**	76.0	78.6
WINN	72.2	76.7	77.3	81.1	77.2	79.9	71.8	76.4	71.4	75.8
RQE LINEAR	**74.5**	**78.8**	**80.3**	**82.8**	**79.8**	81.9	**75.8**	79.3	**76.9**	**78.7**
RQE NONLINEAR	**74.5**	78.6	**80.2**	**82.8**	**79.9**	**82.1**	75.7	**79.5**	**77.0**	**78.7**

TABLE I
DECISION STUMPS (DS) / DECISION TREES (DT): OVERALL CLASSIFICATION ACCURACIES (IN %) AFTER CROSS-VALIDATION AND 500 BOOSTING ITERATIONS. BEST SCORES ARE BOLD, UNDERLINED, SECOND-BEST SCORES BOLD, AND THIRD-BEST SCORES UNDERLINED.

15 × 15 neighborhood. It seems that banks of few specifically predesigned filters decrease performance by artificially constraining the possible solution space of the learner.

The full power of the learning method becomes apparent if we consider the performance of raw pixel intensities within a 15 × 15 neighborhood (displayed in purple in Fig. 7). Although RQE features outperform this baseline (if averaged across all data sets), 15 × 15 results are very close (within two percent points). Rankings for tree boosting (see Fig. 6), which count all data sets separately, show that raw pixel intensities within a 15 × 15 neighborhood rank identically in YUV color space as the linear feature set and very close to the nonlinear feature set. If one wants to avoid the high memory and runtime requirements during the training with RQE, a possibility is to simply use raw pixel intensities within a certain neighborhood (in conjunction with the NDVI, if available, see black curve in Fig. 7, RGB color space). In general, features collected within a certain neighborhood around the pixel of interest always outperform single pixel intensities.

To put the results in our study into a broader context, we

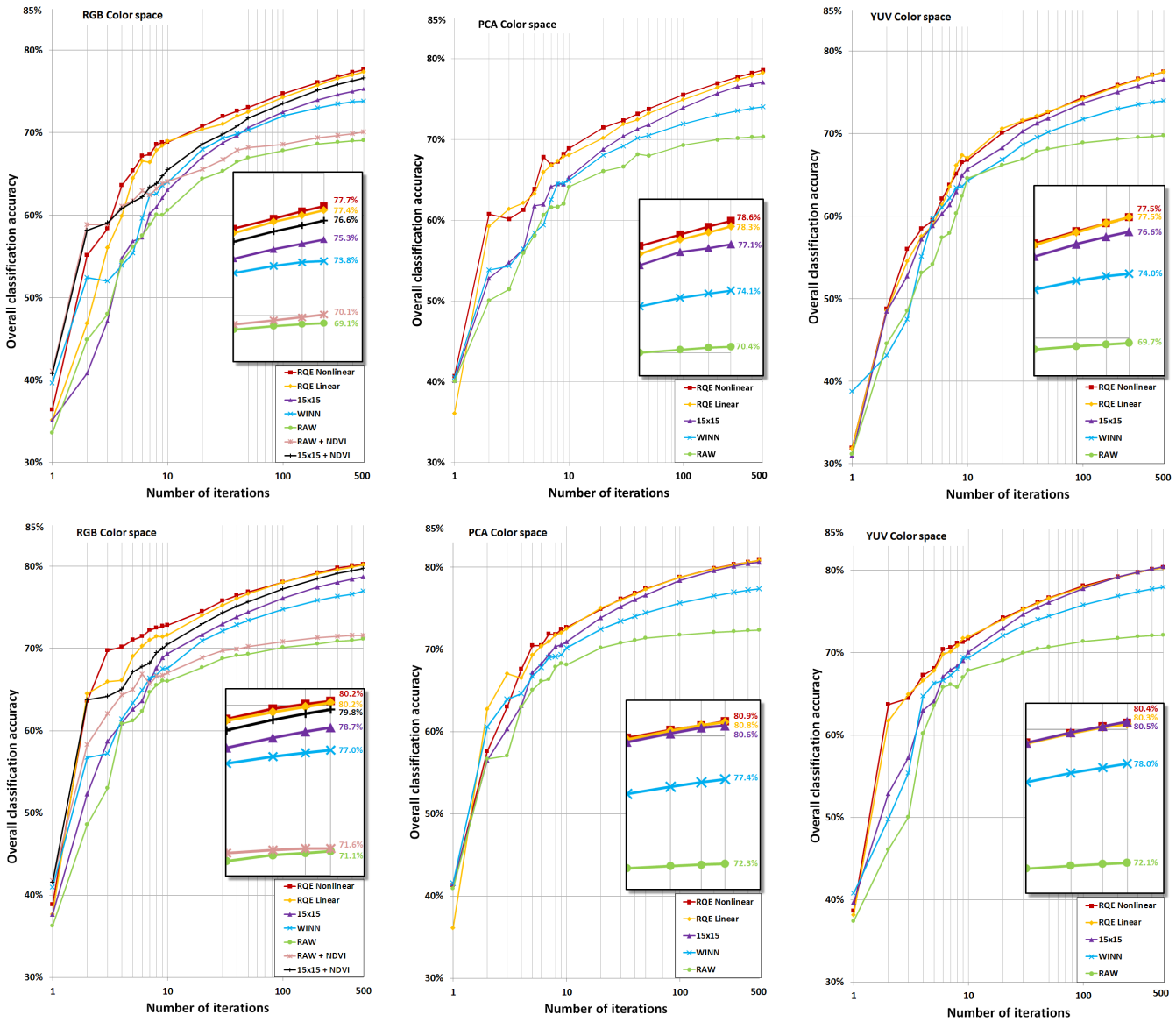

Fig. 7. Overall classification accuracy plotted versus (logarithmically scaled) boosting iterations for stumps (**upper** row) and trees (**lower** row) with features extracted from **RGB, PCA, and YUV** channels.

compare to related work on VHR aerial image classification of urban scenes. For example, Kluckner *et al.* [19] attempt to classify aerial VHR data of Graz, Dallas, and San Francisco into five urban land-cover classes quite similar to the ones we use (buildings, green areas, waterbodies, trees, and street layer). The data consists of standard RGB imagery and the corresponding normalized digital surface model. By combining the covariance descriptors of [56] with a random forest classifier and contextual information modeled by a conditional random field they achieve overall classification accuracies up to 79% (without height information) and 93% (with height information), respectively. Their follow-up work [20], which focusses on improved covariance descriptors, achieves similar accuracies on the same datasets: up to 78% (without height information) and 86% (with height information). In a similar setting Guo *et al.* [21] classify four urban land-cover classes nearly identical to ours (buildings, vegetation, artificial ground, and natural ground) based on VHR aerial images and full-waveform LiDAR data, also using random forests. Overall accuracies range from 82% (using only spectral features) to 96% (spectral and LiDAR features). In that study, the class frequencies of the test data are rather unbalanced, for example, they report only 40% classification accuracy for vegetation, which, however comprises only 4% of the test data. Rottensteiner *et al.* [22] combine LiDAR data with optical features (NDVI) using the Dempster-Shafer theory to detect buildings in urban areas. They achieve 90-95% accuracy.

Here, we obtain accuracies of 79-83% without using height information, which is on par with or slightly above the mentioned state-of-the-art works. It should be pointed out that the

numbers are not really comparable because of large differences between the datasets (e.g. Vaihingen is more difficult than Graz due to its narrow winding roads and small houses), evaluation procedures (e.g. not all authors use cross-validation), class definitions, and amount of training data used.

VI. CONCLUSIONS AND OUTLOOK

We have presented a simple yet powerful strategy for semantic segmentation of VHR remote sensing images. Providing a comprehensive feature bank where the learner directly picks optimal features (implemented efficiently via integral images) leads to very good classification results. A detailed experimental comparison of the proposed RQE feature bank with several baselines using VHR remote sensing images of complex urban scenes underpins this claim. The RQE feature bank in combination with boosting is capable of covering a large range of texture frequencies. Note that no postprocessing or smoothness prior was introduced, which would probably further improve results.

It turns out that data-specific feature engineering seems unnecessary once we directly let the classifier choose the most discriminative features for the data and object classes at hand.

In future work, we plan to add more prior knowledge about the scene structure into the classifier. For example, one could insert this method as unary potential into a conditional random field [57]. Another interesting possibility is to learn object and scene structures via a bag-of-visual-words approach like done for the categorization of scenes in [58].

A general bottleneck of boosting with very large sets of features is the significant computational cost at training time (runtime and memory). In future work, we will thus investigate how we can speed up training and reduce memory requirements. For example, Appel *et al.* [59] recently proposed to accelerate training by pruning irrelevant features early, bounding the split errors early, and only evaluating features to completion that have a chance of becoming the best feature. Another possibility is to parallelize the training of the weak classifiers. The most time-consuming part is finding the best feature to split on in the decision tree nodes; thus, evaluating candidate splits in parallel will result in large speed gains (1-2 orders of magnitude, depending on the number of available processors). That step is trivial to parallelize since the computations are independent.

We believe that, in many remote sensing applications, large general purpose feature sets in combination with discriminative learning can completely relieve the user of choosing features.

REFERENCES

[1] United Nations Population Division, "World Population Prospects 1950-2050. The 2012 Revision. Key Findings and Advance Tables." [Online]. Available: {http://esa.un.org/wpp/Documentation/pdf/WPP2012_%20KEY%20FINDINGS.pdf}

[2] B. Waske and J. Benediktsson, "Fusion of support vector machines for classification of multisensor data," IEEE Transactions on Geoscience and Remote Sensing, vol. 45, no. 12, pp. 3858–3866, 2007.

[3] F. Bovolo, L. Bruzzone, and L. Carlin, "A novel technique for subpixel image classification based on support vector machine," IEEE Transactions on Image Processing, vol. 19, no. 11, pp. 2983 –2999, 2010.

[4] M. Pal, "Random forest classifier for remote sensing classification," International Journal of Remote Sensing, vol. 26, no. 1, pp. 217–222, 2005.

[5] P. O. Gislason, J. A. Benediktsson, and J. R. Sveinsson, "Random forests for land cover classification," Pattern Recognition Letters, vol. 27, no. 4, pp. 294–300, 2006.

[6] G. Briem, J. Benediktsson, and J. Sveinsson, "Multiple Classifiers Applied to Multisource Remote Sensing Data," IEEE Transactions on Geoscience and Remote Sensing, vol. 40, no. 10, pp. 2291–2299, 2002.

[7] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," Science, vol. 313, no. 5786, pp. 504–507, 2006.

[8] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun, "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition," in IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[9] G. Gimel'farb, "Texture Modeling by Multiple Pairwise Pixel Interactions," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 11, pp. 1110–1114, 1996.

[10] S. Zhu, Y. Wu, and D. Mumford, "Minimax Entropy Principle and Its Application to Texture Modeling," Neural Computation, vol. 9, no. 8, pp. 1627–1660, 1997.

[11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[12] P. Simard, L. Bottou, P. Haffner, and Y. LeCun, "Boxlets: a fast convolution algorithm for signal processing and neural networks," in Conference on Neural Information Processing Systems, 1999.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-up robust features (SURF)," Computer Vision and Image Understanding, vol. 110(3), pp. 346–359, 2008.

[14] A. Blum and P. Langley, "Selection of relevant features and examples in machine learning," Artificial Intelligence, vol. 97, pp. 245–271, 1997.

[15] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions," in British Machine Vision Conference, 2002.

[16] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," International Journal of Computer Vision, vol. 81, pp. 2–23, 2009.

[17] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, to appear, available online.

[18] P. Gamba, F. Dell'Acqua, M. Stasolla, G. Trianni, and G. Lisini, Limits and Challenges of Optical Very-High-Spatial-Resolution Satellite Remote Sensing

for Urban Applications. John Wiley & Sons, 2011, pp. 35–48.

[19] S. Kluckner, T. Mauthner, P. Roth, and H. Bischof, "Semantic classification in aerial imagery by integrating appearance and height information," in Computer Vision - ACCV 2009, ser. Lecture Notes in Computer Science, H. Zha, R.-i. Taniguchi, and S. Maybank, Eds. Springer Berlin Heidelberg, 2010, vol. 5995, pp. 477–488.

[20] S. Kluckner and H. Bischof, "Semantic classification by covariance descriptors within a randomized forest," in Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, Sept 2009, pp. 665–672.

[21] L. Guo, N. Chehata, C. Mallet, and S. Boukir, "Relevance of airborne lidar and multispectral image data for urban scene classification using random forests," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 66, no. 1, pp. 56 – 66, 2011.

[22] F. Rottensteiner, J. Trinder, S. Clode, and K. Kubik, "Using the dempster-shafer method for the fusion of {LIDAR} data and multi-spectral images for building detection," Information Fusion, vol. 6, no. 4, pp. 283 – 300, 2005, fusion of Remotely Sensed Data over Urban Areas.

[23] P. Helmholz, C. Becker, U. Breitkopf, T. Bschenfeld, A. Busch, D. Grünreich, S. Müller, J. Ostermann, M. Pahl, F. Rottensteiner, K. Vogt, M. Ziems, and C. Heipke, "Semi-automatic Quality Control of Topographic Data Sets," Photogrammetric Engineering and Remote Sensing, vol. 78, no. 9, pp. 959–972, 2012.

[24] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons," International Journal of Computer Vision, vol. 43, no. 1, pp. 29–44, 2001.

[25] C. Schmid, "Constructing Models for Content-based Image Retrieval," in IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[26] M. Galun, E. Sharon, R. Basri, and A. Brandt, "Texture segmentation by multiscale aggregation of filter responses and shape elements," in IEEE International Conference on Computer Vision, 2003.

[27] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 5, pp. 530 –549, 2004.

[28] G. Lazaridis and M. Petrou, "Image registration using the walsh transform," IEEE Transactions on Image Processing, vol. 15, no. 8, pp. 2343–2357, 2006.

[29] J. Shao and W. Foerstner, "Gabor wavelets for texture edge extraction," in ISPRS Commission III Symposium, 1994.

[30] M. Drauschke and H. Mayer, "Evaluation of texture energies for classification of facade images," in ISPRS Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 38, 2010.

[31] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, "Learning Convolutional Feature Hierarchies for Visual Recognition," in Conference on Neural Information Processing Systems, 2010.

[32] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," in European Conference on Computer Vision, 2010.

[33] ——, "Learning to label aerial images from noisy data," in International Conference on Machine Learning, 2012.

[34] P. Tokarczyk, J. Montoya, and K. Schindler, "An evaluation of feature learning methods for high resolution image classification," ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. I-3, 2012.

[35] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis, "Human detection using partial least squares analysis," in IEEE International Conference on Computer Vision, 2009.

[36] S. Hussain and B. Triggs, "Feature Sets and Dimensionality Reduction for Visual Object Detection," in British Machine Vision Conference, 2010.

[37] F. van Coillie, L. Verbeke, and R. D. Wulf, "Feature selection by genetic algorithms in object-based classification of IKONOS imagery for forest mapping in Flanders, Belgium," Remote Sensing of Environment, vol. 110, pp. 476–487, 2007.

[38] Y. Rezaei, M. Mobasheri, M. V. Zoej, and M. Schaepman, "Endmember Extraction Using a Combination of Orthogonal Projection and Genetic Algorithm," IEEE Geoscience and Remote Sensing Letters, vol. 9, no. 2, pp. 161–165, 2012.

[39] P. Dollár, Z. Tu, H. Tao, and S. Belongie, "Feature Mining for Image Classification," in IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[40] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in British Machine Vision Conference, 2009.

[41] B. Fröhlich, E. Rodner, and J. Denzler, "Semantic Segmentation with Millions of Features: Integrating Multiple Cues in a Combined Random Forest Approach," in Asian Conference on Computer Vision, 2012.

[42] B. Fröhlich, E. Bach, I. Walde, S. Hese, C. Schmullius, and J. Denzler, "Land cover classification of satellite images using contextual information," ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. II(3/W1), pp. 1–6, 2013.

[43] M. J. Mendenhall and E. Merenyi, "Relevance-based feature extraction for hyperspectral images," Neural Networks, IEEE Transactions on, vol. 19, no. 4, pp. 658–672, 2008.

[44] A. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in IEEE International Conference on Computer Vision, 2005.

[45] P. Tokarczyk, J. D. Wegner, S. Walk, and K. Schindler, "Beyond hand-crafted features in remote sensing," ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. II(3/W1), pp. 35–40, 2013.

[46] Y. Freund and R. Schapire, "A decision-theoretic gen-

eralization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119–139, 1997.

[47] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," Machine Learning, vol. 37, no. 3, pp. 297–336, 1999.

[48] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl, "Multiboost: a multi-purpose boosting package," Journal of Machine Learning Research, vol. 13, pp. 549–553, 2012.

[49] C. M. Bishop, Pattern Recognition and Machine Learning. Springer Verlag, 2006.

[50] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning). MIT Press, 2006.

[51] M. Cramer, "The DGPF-test on digital airborne camera evaluation overview and test design," Photogrammetrie – Fernerkundung – Geoinformation, vol. 2, pp. 73–82, 2010.

[52] F. Rottensteiner, G. Sohn, J. Jung, M. Gerke, C. Baillard, S. Benitez, and U. Breitkopf, "The ISPRS benchmark on urban object classification and 3D building reconstruction," in ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. I-3, 2012.

[53] R. C. Gonzalez and R. E. Woods, Digital Image Processing (3rd Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[54] T. Lillesand, R. W. Kiefer, and J. Chipman, Remote Sensing and Image Interpretation. John Wiley & Sons, 2007.

[55] F. Wilcoxon, "Individual Comparisons by Ranking Methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80–83, Dec. 1945.

[56] O. Tuzel, F. Porikli, and P. Meer, "Learning on lie groups for invariant detection and tracking," in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, June 2008, pp. 1–8.

[57] K. Schindler, "An Overview and Comparison of Smooth Labeling Methods for Land-Cover Classification," IEEE Transactions on Geoscience and Remote Sensing, vol. 50, no. 11, pp. 4534–4545, 2012.

[58] A. Cheriyadat, "Unsupervised feature learning for aerial scene classification," IEEE Transactions on Geoscience and Remote Sensing, 2013, to appear, available online.

[59] R. Appel, T. Fuchs, P. Dollár, and P. Perona, "Quickly Boosting Decision Trees - Pruning Underachieving Features Early," in International Conference on Machine Learning, 2013.

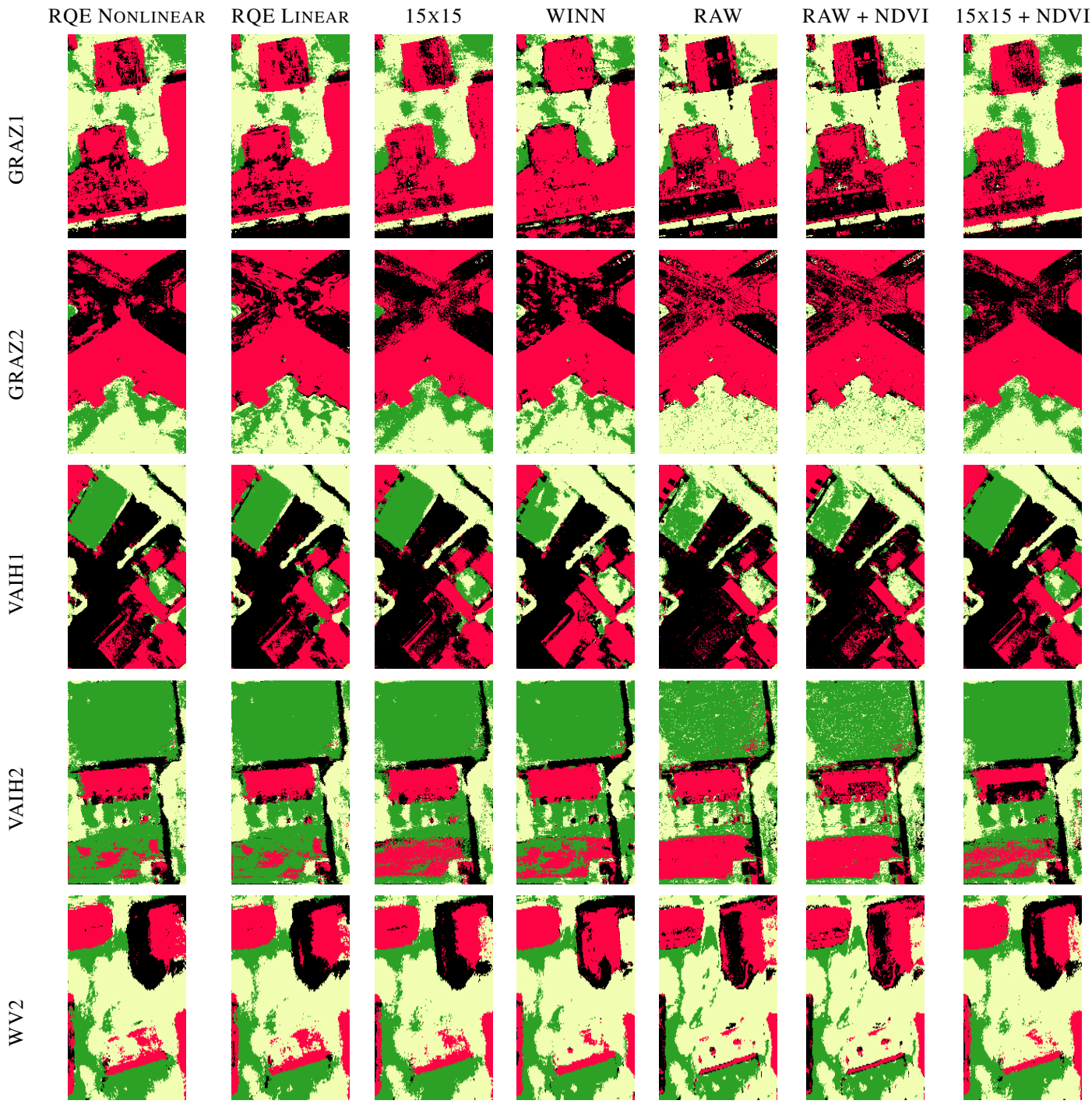

Fig. 8. **RGB** with boosting decision trees: results for the different feature settings of a subscene per original test image. Streets are displayed black, buildings red, high vegetation green, and low vegetation yellow.

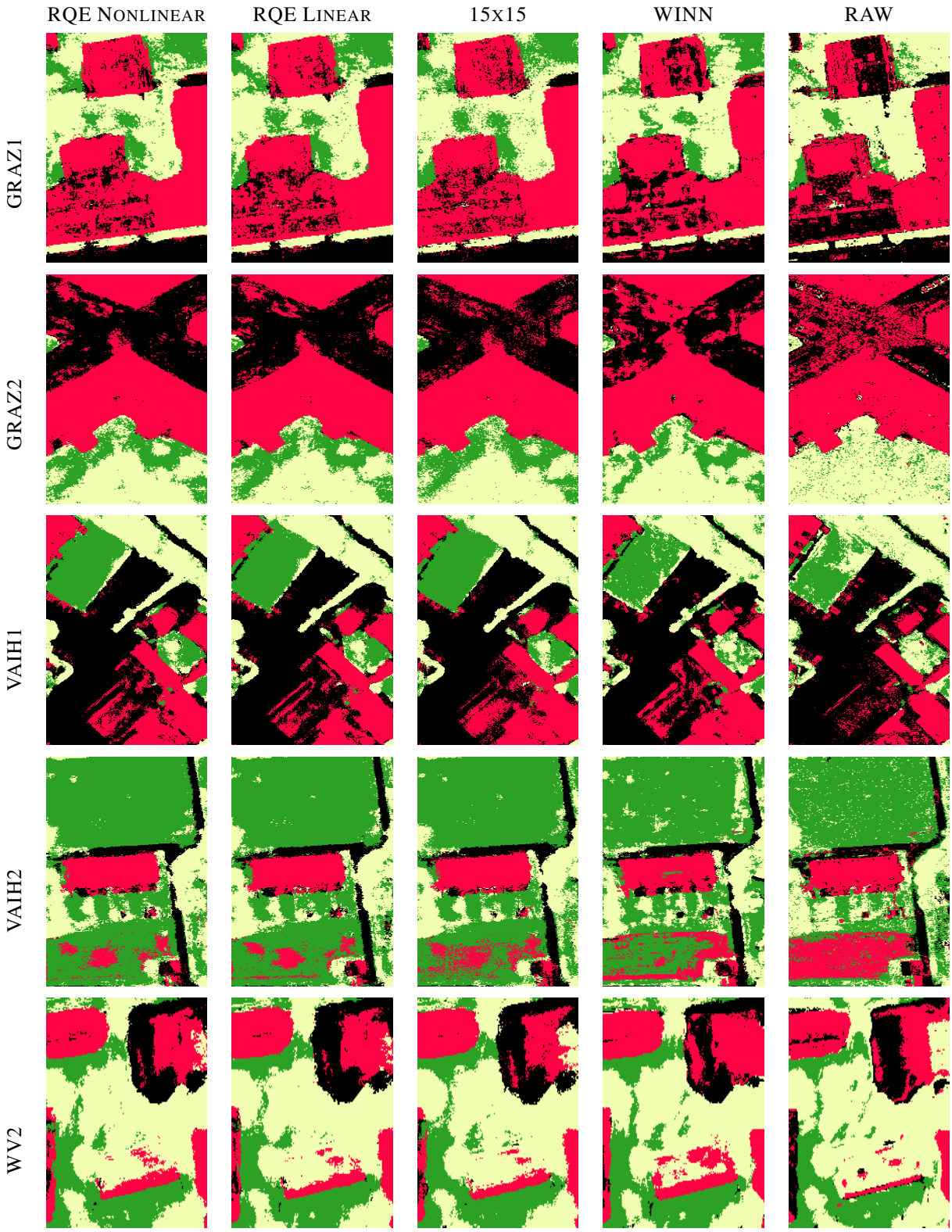

Fig. 9. **PCA** with boosting decision trees: results for the different feature settings of a subscene per original test image. Streets are displayed black, buildings red, high vegetation green, and low vegetation yellow.

Fig. 10. Cutouts of the test images (cf. Fig. 4) with manually labeled ground truth in the bottom row.

FEATURE EXTRACTION [MIN]	GRAZ1	GRAZ2	VAIH1	VAIH2	WV2
RAW	1	2	2	2	2
15×15	14	20	20	23	20
WINN	1	2	2	2	2
RQE NONLINEAR	150	224	222	215	230
TRAINING DECISION STUMPS [MIN]					
RAW	241	404	546	415	490
15×15	5098	7451	8429	8080	7201
WINN	532	887	1180	1026	814
RQE NONLINEAR	5501	8383	11371	9874	9310
TRAINING DECISION TREES [MIN]					
RAW	511	898	1222	991	948
15×15	15702	26008	33556	27405	25548
WINN	1443	2373	2896	2666	2137
RQE NONLINEAR	19245	29297	36689	34708	32192
TESTING [MIN]					
RAW	6	9	9	9	9
15×15	13	17	18	18	17
WINN	7	10	11	11	10
RQE NONLINEAR	70	85	84	83	86

TABLE II
FEATURE EXTRACTION, TRAINING DECISION STUMPS, TRAINING DECISION TREES, TESTING CPU TIMES (IN MINUTES) NEEDED FOR EXTRACTING AND RECORDING DIFFERENT FEATURES. THE EVALUATION HAS BEEN PERFORMED ON THE AMD OPTERON CPUS AND 32/64 GB OF RAM.

VAIH2, RGB, RQE Nonlinear	FEATURE SUB-GROUP	OCCURENCE	SUM OF WEIGHTS
FIRST 10 ITERATIONS	Random patches	1	0.1730
	Centered patches	0	0.0000
	Centered patches, different channels	1	0.1996
	Centered patches, different sizes	0	0.0000
	Single pixel values	0	0.0000
	Pixel values after 3×3 mean filtering	0	0.0000
	Pixel values after 5×5 mean filtering	7	1.3675
	Central nonlinear patches	0	0.0000
	Central nonlinear patches, different sizes	1	0.5455
FIRST 100 ITERATIONS	Random patches	12	0.9665
	Centered patches	0	0.0000
	Centered patches, different channels	6	0.5617
	Centered patches, different sizes	1	0.0535
	Single pixel values	15	1.0382
	Pixel values after 3×3 mean filtering	17	1.2035
	Pixel values after 5×5 mean filtering	24	2.5193
	Central nonlinear patches	4	0.2528
	Central nonlinear patches, different sizes	21	2.0341
FIRST 500 ITERATIONS	Random patches	81	2.9499
	Centered patches	2	0.0663
	Centered patches, different channels	33	1.4314
	Centered patches, different sizes	17	0.4876
	Single pixel values	146	4.8468
	Pixel values after 3×3 mean filtering	50	2.1927
	Pixel values after 5×5 mean filtering	90	4.4649
	Central nonlinear patches	19	0.6920
	Central nonlinear patches, different sizes	62	3.3343

TABLE III

VAIH2, RGB, RQE NONLINEAR: A LIST OF THE FEATURE TYPES THAT HAVE BEEN CHOSEN DURING TRAINING THE BOOSTING CLASSIFIER (DECISION STUMPS SETTING) AFTER 10, 100 AND 500 ITERATIONS. WE SHOW THE FREQUENCY OF A PARTICULAR GROUP BEING CHOSEN AND A SUM OF THE WEIGHTS GIVEN TO A SPECIFIC FEATURE GROUP BY THE CLASSIFIER.

		RGB								PCA								YUV							
		BUI		STR		GRA		TRE		BUI		STR		GRA		TRE		BUI		STR		GRA		TRE	
		UA	PA	UA	PA	UA	PA	UA	PA	UA	PA	UA	PA	UA	PA	UA	PA	UA	PA	UA	PA	UA	PA	UA	PA
GRAZ1	RAW	77.5	78.4	60.5	62.3	52.7	27.1	69.4	82.6	80.1	79.1	63.6	67.2	53.1	29.6	70.0	**82.7**	80.2	78.0	62.5	68.0	52.2	28.5	69.7	82.3
	RAW+NDVI	79.6	78.9	62.7	66.5	52.6	28.0	69.9	81.8	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15×15	80.4	81.3	67.4	68.9	**65.7**	47.8	**77.0**	**84.4**	84.5	85.3	74.0	75.2	**63.4**	**49.3**	**76.4**	**82.5**	**84.3**	**85.4**	**74.7**	**74.7**	**63.6**	**49.0**	**77.3**	**82.5**
	15×15+NDVI	82.8	83.5	72.0	73.7	59.6	43.2	73.2	79.5	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	WINN	83.0	**84.4**	69.2	69.9	53.7	43.1	72.3	80.7	81.1	83.1	69.5	69.0	60.6	47.4	75.7	81.2	82.8	83.5	70.5	72.1	59.5	45.6	76.0	81.0
	RQE LINEAR	**83.5**	**84.0**	**72.5**	**73.8**	**62.7**	**48.0**	76.9	**83.1**	**85.2**	**85.9**	**75.5**	**76.5**	**60.9**	**49.8**	**76.6**	80.8	**84.2**	85.1	**74.3**	**75.1**	**62.1**	**48.2**	**76.6**	81.3
	RQE NONLIN	**83.8**	83.8	**72.3**	**74.4**	62.5	**48.8**	**77.0**	82.6	**85.7**	**86.1**	**75.7**	**77.3**	60.3	47.6	76.3	81.5	84.0	**85.2**	74.2	74.6	61.3	47.1	76.3	**81.4**
GRAZ2	RAW	83.5	84.6	69.7	69.3	49.0	14.1	63.9	**88.1**	85.0	84.2	70.2	72.5	48.7	16.9	64.2	**86.7**	84.6	83.7	69.5	72.1	49.3	16.6	64.5	**87.1**
	RAW+NDVI	85.1	83.5	69.1	73.2	48.6	16.9	64.8	**86.5**	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15×15	87.0	87.0	77.6	77.3	**69.6**	**61.7**	78.3	83.1	**89.9**	88.7	80.5	**82.7**	**67.1**	60.1	77.6	**82.2**	**89.2**	88.6	79.9	**81.3**	**67.6**	**61.0**	**78.3**	**82.2**
	15×15+NDVI	**89.0**	88.0	**79.3**	**81.2**	68.5	**61.7**	**78.8**	82.4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	WINN	87.2	86.8	76.1	77.4	65.8	56.1	76.3	81.3	87.8	87.2	77.5	78.6	65.8	57.6	76.8	81.7	88.0	87.7	77.7	79.0	65.4	56.7	76.7	81.6
	RQE LINEAR	87.2	**88.2**	78.8	77.6	66.5	58.5	77.4	82.1	**89.7**	**89.2**	**81.2**	**82.7**	**67.5**	**60.9**	**78.3**	**82.2**	**89.2**	**88.9**	**80.3**	**81.3**	66.8	**61.1**	**78.3**	81.7
	RQE NONLIN	**88.6**	**88.3**	**79.8**	**80.5**	**69.0**	**62.6**	**79.3**	83.0	**89.7**	**89.1**	**81.2**	82.5	66.9	**60.5**	**77.9**	82.1	**89.2**	**89.0**	**80.5**	**81.4**	**67.0**	60.4	78.1	81.9
VAIH1	RAW	77.7	69.4	74.6	83.1	53.6	45.8	78.2	84.7	80.2	69.8	75.1	84.6	54.4	47.1	78.5	84.5	79.9	70.7	75.2	84.3	54.5	47.1	78.6	84.4
	RAW+NDVI	78.3	69.2	74.3	83.0	53.5	46.0	78.2	84.5	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15×15	80.9	79.9	81.3	84.5	**66.8**	64.3	**84.6**	**88.0**	**82.6**	**81.6**	**83.3**	**85.2**	**66.8**	65.7	**84.9**	**88.2**	**82.5**	**81.5**	**83.0**	**85.0**	**66.3**	**64.8**	**84.3**	**88.1**
	15×15+NDVI	81.0	**80.9**	**82.3**	83.9	**67.1**	**66.4**	**84.8**	**88.1**	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	WINN	**83.2**	79.0	80.8	**85.2**	59.3	55.4	79.8	86.3	81.8	79.4	81.1	84.9	57.1	52.0	79.8	84.7	81.7	78.7	80.8	**84.9**	61.2	57.7	82.2	86.5
	RQE LINEAR	**82.9**	77.5	81.5	84.9	61.8	64.0	84.5	83.9	**82.9**	**81.4**	**83.2**	**85.5**	**67.2**	**66.6**	**84.6**	**88.0**	81.9	81.4	82.5	84.6	65.7	64.3	**84.3**	**87.6**
	RQE NONLIN	82.7	**80.4**	**82.3**	**85.3**	65.8	**65.1**	84.5	87.9	82.4	81.1	83.1	85.0	66.6	**66.4**	**84.6**	87.9	**82.2**	**81.7**	**82.7**	84.8	**65.9**	**65.0**	**84.5**	**87.6**
VAIH2	RAW	68.8	70.4	65.5	69.1	59.9	57.2	74.0	72.5	71.0	72.4	67.2	69.8	62.7	56.9	72.6	74.9	71.0	72.6	67.5	70.3	61.8	56.7	72.9	74.5
	RAW+NDVI	69.7	70.5	65.6	69.1	59.3	57.2	73.6	72.4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15×15	79.2	79.4	73.8	77.5	76.0	69.7	**80.5**	**82.1**	82.7	83.2	**77.5**	**78.9**	75.6	71.3	**79.7**	81.0	81.4	**83.0**	**78.2**	**79.1**	**75.4**	71.2	**80.1**	**81.1**
	15×15+NDVI	79.6	80.3	74.2	77.2	76.2	70.3	80.2	81.7	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	WINN	77.3	77.7	73.2	74.1	70.7	65.1	76.3	80.5	78.4	79.4	73.9	76.8	68.1	61.1	74.2	77.3	79.1	78.6	73.9	76.5	72.0	66.7	77.6	79.9
	RQE LINEAR	**82.2**	**81.8**	**75.6**	**77.7**	**76.6**	**72.3**	**80.6**	**82.3**	**83.3**	**83.3**	77.3	**78.9**	**76.1**	**71.5**	79.4	**81.4**	**82.0**	**83.0**	77.4	**78.9**	75.0	**71.6**	**80.1**	80.5
	RQE NONLIN	**82.2**	**81.8**	**75.3**	**77.6**	**76.7**	**72.3**	80.4	82.0	**83.4**	**83.6**	**77.8**	**79.2**	**76.1**	**71.7**	**79.6**	**81.3**	**82.1**	**83.3**	**77.6**	78.8	**75.3**	**71.7**	**80.1**	**80.9**
WV2	RAW	69.7	68.6	57.4	54.0	78.3	71.5	72.2	81.2	70.6	69.9	59.0	55.6	78.6	71.6	72.1	80.7	70.1	68.5	57.6	54.4	**79.2**	71.1	71.8	81.8
	RAW+NDVI	69.7	68.9	57.4	53.7	78.4	70.8	71.8	81.1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15×15	81.7	82.1	72.7	**71.0**	**79.8**	75.5	78.3	**82.1**	81.4	82.7	73.7	70.9	**79.4**	**77.9**	**79.2**	**81.2**	81.5	81.3	**72.4**	**70.4**	**79.7**	77.4	79.1	**82.4**
	15×15+NDVI	**81.9**	**82.4**	**73.0**	**71.0**	**79.7**	**76.2**	**78.6**	**82.1**	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	WINN	78.4	78.5	69.7	67.3	78.3	74.7	77.2	80.9	79.2	80.6	71.0	68.6	78.9	76.8	78.7	80.7	78.2	78.9	69.5	66.2	78.1	74.2	76.3	81.3
	RQE LINEAR	**81.9**	**82.4**	72.7	**71.0**	79.6	**76.2**	**78.6**	81.7	**81.8**	**82.8**	**73.8**	**71.1**	**79.1**	77.5	**79.0**	**81.3**	**81.9**	**82.0**	**72.4**	**70.3**	**79.2**	**77.7**	**79.3**	**81.9**
	RQE NONLIN	**81.9**	**82.7**	**73.1**	**71.3**	**79.7**	**76.6**	**78.7**	81.5	**81.5**	**83.0**	**73.9**	**71.0**	79.0	**77.6**	**79.0**	81.1	**81.9**	**82.2**	**72.8**	70.1	78.7	**78.1**	**79.5**	81.5

TABLE IV

DECISION TREES: USER'S (UA) AND PRODUCER'S (PA) ACCURACIES (IN %) AFTER CROSS-VALIDATION AND 500 BOOSTING ITERATIONS. BUI STANDS FOR BUILDINGS, STR FOR STREETS, GRA FOR GRASS, AND TRE FOR TREES. BEST SCORES ARE BOLD, UNDERLINED, SECOND-BEST SCORES BOLD, AND THIRD-BEST SCORES UNDERLINED.