On-line Boosting for Car Detection from Aerial Images

Thuy Thi Nguyen, *Student Member, IEEE*, Helmut Grabner, *Student Member, IEEE*, Barbara Gruber and Horst Bischof, *Member, IEEE*

***Abstract*— In this paper, we present a new approach for automatic car detection from aerial images. The system exploits a robust machine learning method known as boosting for efficient car detection from high resolution aerial images. We propose to use on-line boosting with interactive training framework to efficiently train and improve the detector. We use integral images for fast computation of features. This also allows to perform exhaustive search for detection of cars after training. For post processing, we employ a mean shift clustering method, which improves the detection rate significantly. In contrast to related work, our framework does not rely on any priori knowledge of the image like a site-model or contextual information, but if necessary this information can be incorporated. An extensive set of experiments on high resolution aerial images using the new UltraCamD shows the superiority of our approach.**

***Index Terms*— Machine learning, pattern recognition, Adaboost, on-line learning, computer vision, object detection, car detection, aerial image, UltraCamD.**

I. Introduction

RECENT years, a robust machine learning method named boosting has become popular. Boosting has been used for text recognition, text filtering, routing, "ranking" problems, learning problems in natural language processing, medical diagnostic, customer monitoring and segmentation [33]. Various boosting frameworks have been developed for solving machine learning problems [33], [7], [8], [37]. Following the remarkable success of the face detector introduced by Viola and Jones in [39], boosting techniques have been widely used for different problems in the computer vision community. The detection problem is formulated as binary classification problem, discriminating the object from the background. The learned classifier is evaluated on the whole image. In order to speed up the exhaustive search in the classical work of [39], integral images were employed, which allow very fast computing of simple image features for object representation. Besides, a cascade structure enables the detector to be simultaneously fast and accurate. This framework allows to proceed efficiently on large image data and has been successfully applied in various object detection problems. Some on-line learning frameworks have been proposed to deal with this problem for object detection and classification [14], [18], [22], [30], [1]. The on-line strategy aims to reduce hand labeling effort of training samples and gives possibilities to increase variances of training data in an on-line manner, while progressively improving the classifier.

url: http://www.rivf.org. Email: rivf@rivf.org
Supported by IEEE Region 10

Building an efficient and robust framework for object detection from in aerial images has drawn the attention of research community in computer vision for years, e.g. [32], [29], [44], [12], [3]. The problem of car detection from aerial images has a variety of civil and military applications, for example transportation control, road verification to complete land use classification problem for urban planning, or military reconnaissance, etc.

An aerial image taken by airplane is a large-scale image which contains a lot of objects with a complicated background of the urban scene. For example, *UltraCamD* camera from Microsoft-Vexcel can deliver large format panchromatic images as well as multi spectral images, see [16]. The high resolution images have a size of 11500 pixels across-track and 7500 pixels along-track. Thus a panchromatic image has a size of 84 MB and a RGB or NIR (near infrared) image has a size of 252 MB. These large images need automatic methods for efficient processing. Besides, aerial images are usually taken from a vertical or slightly slanting direction. Although with some constraints on the viewpoint, the appearance of the cars in the image is widely varying. Cars appear as small objects, which vary in intensity and many details are not visible. Moreover, the urban scene contains a complicated background with variety of objects that look like cars such as windows, roofs of buildings, corners of streets. All these properties make it difficult to characterize the features of a car and imposes challenges in recognizing cars from aerial images.

The approaches to object (car) detection can be considered as two groups according to the type of the used models: the explicit modeling approach and the implicit modeling approach [12].

The explicit model approach uses a generic car model [44], [21], [12], [35], [13]. A car is represented as a 2D or 3D model of the shape of cars. Features are prominent geometric features of cars on different levels of detail. In the detection stage, image features are extracted and grouped to construct structures similar to a car model. The car object is considered to be detected if there is sufficient evidence of the model in the image. This approach relies mainly on geometric features with edges, lines and surfaces to construct a hierarchical structure. In aerial images cars are rather small, therefore the models can not be very detailed because the features are no longer detectable, on the other hand generic and simple models have the inherent danger of fitting to many places on the image, therefore not being discriminative enough.

In implicit or appearance-based approach, the car model is created by example images of cars, which consist of gray

value or texture features. Appearance models are created by collecting statistics over these features. For car detection in terrestrial images, some part or component based models have been proposed [5], [11], [4]. The classifier architecture can be a single classifier, a classifier combination or a hierarchical model. Support vector machines were mainly used for classification [29], [36], [26], [11], [4]. The detection is done for image regions by computing the feature vectors and classifying them against the model features. Although these approaches have certain advantages there exist drawbacks. Mostly, the feature's calculation and classification is computational expensive. Moreover, there is a need of huge amount of labeled data for training the detector. The training set should provide a good coverage over the space of possible appearance variations of the data. This needs a lot of time and labor forces to build the training data in advance and limits the possibility to diversify the variances of training samples during the training phase.

None of the mentioned work (up to our knowledge) uses boosting methods for object (car) detection from aerial images. In this paper, we focus on developing a robust boosting-based system for car detection from aerial images. The main goal is to obtain a high quality car detection system by using novel machine learning methods with an efficient training mechanism.

First, we propose to use boosting on aerial images. In particular, we use efficient integral image representation for fast calculation of car's features. In addition to the commonly used Haar wavelet [39], we use local orientation histogram [6] and local binary patterns [23] as features.

Second, a novel on-line version of Adaboost is employed to train the detector. The algorithm performs on-line updating of the ensembles of features during the training process. By on-line training, we can update the classifier as a new sample arrives, and therefore can minimize the tedious work of hand labeling of training samples.

The developed framework results in a robust and automatic car detection system from aerial images and achieves a high performance. The system is flexible since we do not use any constraint to site-model or textual knowledge or other information about the appearance of cars in the images.

The paper is organized as the following. Section II presents our approach for building the framework for car detection from aerial images. Section III is dedicated to experiments and results. An additional discussion about the utility of all the available data delivered by UltraCamD to integrate our system with related applications is also presented. Section IV is for discussion and future work.

II. System description

We propose an on-line boosting based framework for car detection from aerial images based on appearance models. First, we summarize the boosting method which will be used for feature selection. The active training process, which allows efficient on-line learning, is described afterward. Then, the features used for classification are discussed. Car detection is performed on an image by applying the trained classifier in an exhaustive search over all possible locations and rotations of the image. Finally, a post processing stage using the mean shift clustering technique is presented to improve detection rate. In addition, we show how context information can be used to further improve the detection results.

A. Boosting

In general boosting converts (boosts) a weak learning algorithm into a strong one. Boosting has been analyzed carefully (e.g. [34]) and tested empirically by many researchers. Various variants of Boosting have been developed (e.g. Real-Boost [8], LP-Boost [7]). We focus on the discrete AdaBoost (adaptive boosting) algorithm introduced by Freund and Shapire [8]. It adaptively re-weights the training samples instead of re-sampling them.

The basic algorithm works as follows: Given a training set $\mathcal{X} = \{\langle \mathbf{x_1}, y_1\rangle, ..., \langle \mathbf{x_L}, y_L\rangle \mid \mathbf{x_i} \in \mathbf{R}^m,\ y_i \in \{-1, +1\}\}$ with positive and negative labeled samples and an initial uniform distribution $p(\mathbf{x_i}) = \frac{1}{L}$ over the examples.

Based on $\mathcal{X}$ and $p(\mathbf{x})$ a weak classifier h^{weak} is trained. A weak classifiers is a classifier that has to perform only slightly better than random guessing, i.e., for a binary decision task, the error rate must be less than 50%. The classifier is obtained by applying a learning algorithm (e.g. applying statistical learning for a decision stump). Based on the error e_n the weak classifier h_n^{weak} gets assigned a weight $\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-e_n}{e_n}\right)$. The probability $p(\mathbf{x})$ is updated such that it increases for the samples that are misclassified. If the sample is classified correctly the corresponding weight is decreased. Therefore, the algorithm focuses on the difficult examples. The process is repeated, and at each boosting iteration a new weak classifier is added, until a certain stopping condition is met (e.g. a given number of weak classifiers are trained).

Finally, a strong classifier $h^{strong}(\mathbf{x})$ is computed as linear combination of a set of N weak classifiers $h_n^{weak}(\mathbf{x})$:

$$h^{strong}(\mathbf{x}) = \text{sign}(conf(\mathbf{x})) \quad (1)$$

$$conf(\mathbf{x}) = \frac{\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x})}{\sum_{n=1}^{N} \alpha_n} \quad (2)$$

As $conf(\mathbf{x})$ is bounded by $[-1, 1]$, it can be interpreted as a confidence measure. The higher the absolute value is, the more confident is the result.

Freund and Schapire [8] proved strong bounds on the training and generalization error of AdaBoost. For the case of binary classification the training error drops exponentially fast with respect to the number of boosting rounds N (i.e. number of weak classifiers). Schapire et al. [34], [31] showed that boosting maximizes the margin and proved that larger margins for the training set are translated to superior upper bounds on the generalization error.

B. On-line Boosting for feature selection

Boosting for feature selection was first introduced by Tieu and Viola [38]. Feature selection from a large set of features is done by Adaboost. The main idea is that each feature corresponds to a single weak classifier and boosting selects an informative subset from these features.

Training proceeds similar to the described boosting algorithm. Given a set of possible features $\mathcal{F} = \{f_1, ..., f_k\}$ in each iteration step n the algorithm builds a weak hypothesis based on the weighted training samples. The best one forms the weak hypothesis h_n^{weak} which corresponds to the selected feature f_n. With respect to the error of the chosen hypotheses the weights of the training samples are updated. Finally, a strong classifier h^{strong} is computed as a weighted linear combination of the weak classifiers, where the weights α_n are estimated according to the errors of h_n^{weak} as described above. Boosting for feature selection as described above works off-line. Thus, all training samples must be given in advance. In our work we use on-line feature selection [10] based on an on-line version of AdaBoost [25], [24]. Therefore, each boosting step of the off-line algorithm has to be done on-line. The mechanism performs an update of weak classifiers whenever a new training sample arrives, which allows to adaptively train the detector and efficiently generate the training set. First, we briefly summarize the on-line boosting idea. Second, we discuss how it can be used for feature selection.

The basic idea in on-line boosting is that the importance (difficulty) of a sample can be estimated by propagating it through the set of weak classifiers. One can think of this, as modeling the information gain with respect to the first n classifier and code it by the importance weight λ (initialized by 1) for doing the update of the $n+1$-th weak classifier. Oza [25] has proved, if off-line and on-line boosting are given the same training set, then the weak classifiers returned by on-line boosting converges statistically to the one obtained by off-line boosting as the number of iterations $N \rightarrow \infty$. Therefore, for repeated presentation of the training set on-line boosting and off-line boosting deliver the same result.

In our system, on-line boosting for feature selection is performed by introducing "selectors" and perform on-line boosting on these selectors and not directly on the weak classifiers. Each selector $h^{sel}(\mathbf{x})$ holds a set of M weak classifiers $\{h_1^{weak}(\mathbf{x}), \ldots, h_M^{weak}(\mathbf{x})\}$ and selects one of them

$$h^{sel}(\mathbf{x}) = h_m^{weak}(\mathbf{x}) \tag{3}$$

according to an optimization criterion (we use the estimated error e_i of each weak classifier h_i^{weak} such that $m = \arg\min_i e_i$).

Note, that the selector can be interpreted as a classifier (he switches between the weak classifiers). Training a selector means that each weak classifier is trained (updated) and the best one (with the lowest estimated error) is selected. Similar to the off-line case, the weak classifiers correspond to features, i.e. the hypotheses generated by the weak classifier is based on the response of the feature.

In particular, the on-line training framework of AdaBoost for feature selection works as follows: First, a fixed set of N selectors $h_1^{sel}, .., h_N^{sel}$ is initialized randomly with weak classifier (i.e. features). When a new training sample $\langle \mathbf{x}, y \rangle$ arrives the selectors are updated. This update is done with respect to the importance weight λ of the current sample. For updating the weak classifiers, any on-line learning algorithm can be used (see Section II-C for more details). The weak classifier with the smallest estimated error is selected by the selector. Finally, the corresponding voting weight α_n and the importance weight λ of the sample are updated and passed to the next selector h_{n+1}^{sel}. The weight increases if the example is misclassified by the current selector or decreased otherwise. For more details see [10].

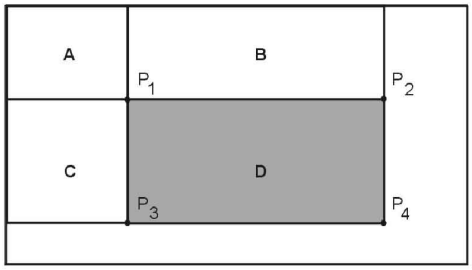


Fig. 1. Efficient calculation of the sum of a rectangular area. The value of the integral image at Position P_1 is the sum of the pixel values in region A. P_2 corresponds $A+B$, P_3 to $A+C$ and P_4 to $A+B+C+D$. Therefore, the sum of the area D can be calculated by only applying 4 reference points as $P_4 + P_1 - P_2 - P_3$.

Finally, a strong classifier is obtained by linear combination of N selectors.

$$h^{strong}(\mathbf{x}) = \text{sign}\Big(\sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(\mathbf{x})\Big) \tag{4}$$

In contrast to the off-line version a classifier is available at any time.

C. Image representation and features

The main purpose of using features instead of raw pixel values as input to a learning algorithm is to reduce the inter-class variability while increasing the out-of-class variability. In addition "ad-hoc" knowledge can be included. In our work we use three different types of features, which are: Haar-like features [39], Orientation histograms [17], [6] and a simple version of local binary patterns (LBP) [23]. One can think of combining more of such (local) features and also include global features, like in [43].

Note, that the computation of all feature types can be done very efficiently using integral images [39] and integral histograms [28] as data structures. This allows to do exhaustive template matching when scanning the whole image. An integral image II sums up all the pixel values from the upper left up to the current position, more formally it is defined on an image I as

$$II(x, y) = \sum_{x'=1}^{x} \sum_{y'=1}^{y} I(x', y') \tag{5}$$

This pre-calculcation of an integral image can be efficiently implemented in one pass over the image. Afterwards, any sum of any rectangular region can be computed by only 4 memory accesses and 3 additions, see

1 for an example. This idea can be easily extended to represent histograms: For each bin one integral image is built separately.

Since we know the resolution of the image, search for cars at different scales is not necessary, but cars can appear at any

orientation. Instead of training the classifier with different orientations we train it at one "norm" orientation, and evaluate it by rotating the detector. Rotation of the detector can be done by computing the features at different angles for the detection process. Lienhart [19] introduced an additional set of rotated Haar-like features, which are an enriched set of basic features and can be computed efficiently. [20] proposed to use different types of Haar-like features. A previously trained classifier is converted to work at any angle, so rotated objects can be detected. A real-time version for the rotational invariant Viola-Jones detector has been reported in [40]. A similar technique is employed in our system, the detector is rotated by increments in $10°$. The rotation for the orientation histogram features can be done very easy by shifting the histogram.

To obtain a weak classifier h_j^{weak} from a feature j, where $f_j(\mathbf{x})$ evaluates this feature on the image $\mathbf{x}$ we model the probability distribution of this feature for positive and negative samples. Following [10] we incrementally estimate the probability $P(1|f_j(\mathbf{x}))$ by assuming a Gauss distribution $\mathcal{N}(\mu^+, \sigma^+)$ (i.e. we incrementally update μ^+ and σ^+) for positive labeled samples and $P(-1|f_j(\mathbf{x}))$ by $\mathcal{N}(\mu^-, \sigma^-)$ for negative labeled samples.

For the classic Haar Wavelets we use a Bayesian decision criterion based on the estimated Gaussian probability density function $g(x|\mu, \sigma)$.

$$h_j^{weak}(\mathbf{x}) = \text{sign}(P(1|f_j(\mathbf{x})) - P(-1|f_j(\mathbf{x}))) \quad (6)$$

$$\approx \text{sign}(g(f_j(\mathbf{x}|\mu^+, \sigma^+) - g(f_j(\mathbf{x})|\mu^-, \sigma^-)) \quad (7)$$

For the histogram based feature types (orientation histograms and LBP), we use a nearest neighbor learning algorithm. The positive and negative samples are modeled by one cluster each. The cluster center $\mathbf{p_j}$ and $\mathbf{n_j}$ are incrementally updated. The weak classifier is given by

$$h_j^{weak}(\mathbf{x}) = \text{sign}(D(f_j(\mathbf{x}), \mathbf{p_j}) - D(f_j(\mathbf{x}), \mathbf{n_j})) \quad (8)$$

where D is a distance metric, in our case the Euclidian norm is used.

Of course, other types and other learning algorithms can be used to obtain a weak hypotheses.

D. Training and detection

The training process is performed by iteratively labeling samples from the images and updating parameters for the model. The labeled samples can be positive or negative. In order to minimize the hand labeling effort we apply an active learning strategy. The key idea is that the user has to label only examples which are not classified correctly by the current classifier. In fact, it has been shown in the active learning community [27], that it is more effective to sample at the current estimate of the decision boundary than the unknown true boundary. This is exactly achieved by our approach. We first evaluate the current classifier on an image. The human supervisor labels additionally "informative" samples, e.g. mark the wrongly labeled examples (i.e. either a false detection or a missing one) and performs an update of the classifier. The new classifier is applied again on a new (or the same) image and the process continues. This is a fully supervised and interactive process. The sketch of the on-line training process for car detection as following:

Algorithm 1 On-line training process

```
Initialize parameters for the classifier
while non-stop-criteria do
    Evaluate the current classifier and display results
    Manually label one "good" sample (either positive or
    negative)
    Update parameters for the classifier
end while
```

The classifier is evaluated and updated after labeling each sample. Since labeling of samples in training phase is an interactive process with visualization, we can intuitively choose to label the most informative and discriminative sample at each update. This allows the parameters of the model to be updated in a greedy manner with respect to minimizing the detection error, meaning that the parameters of the model can be learned very fast. This process avoids labeling redundant samples that do not contribute to the current decision boundery.

After training, detection is performed by applying the trained classifier exhaustively on the images. A car is considered to be detected if the output confidence value of the classifier is above a threshold (i.e. zero). The lower the threshold the more likely an object is detected as a car but on the other hand the more likely a false positive occurs. For a higher threshold the false positives decreases at the expense of the detections. This process delivers many overlapping detections, which are the probabilities of the appearance of an object at a certain location. Therefore a post processing stage is needed to refine and combine these outputs, which significantly improves the detection rate.

E. Post processing

Following the work of [9] we use a non-parametric clustering-based object detection derived from the distribution of classifier output probabilities. The strong classifier generates a probabilistic output. For each image location U we obtain multiple outputs P_k representing object probabilities (in our case the confidence $conf(\cdot)$ of the strong classifier) at each angle k of the image. To obtain a distribution of object probabilities at each rotation, we apply kernel density estimation. Let $\{U_i\}_{i=1,\ldots,n}$ denote the image locations where classification is performed. For each angle k we obtain a probability density estimate

$$\hat{p}_k(\mathbf{u}) = \sum_{i=1}^{n} P_k(U_i) \cdot K_k\left(\frac{\mathbf{u} - U_i}{W}\right), \quad (9)$$

where K_k is two-dimensional Gaussian kernel with a size equivalent to the object size W and scaled by the confidence of the classifier output. The derivative of the probability density distribution is denoted as $\hat{p}_c(\mathbf{u}) = \sum_k \hat{p}_k(\mathbf{u})$. It corresponds to a cumulative density estimate containing the

sum of probabilities over all angles. Mean shift clustering is applied to this density estimate to delineate the appearance of objects. In our case, a simple version is used where K is two-dimensional flat kernel.

F. Land Use Classification and Street Layer

In some applications the context information of aerial image is given and can be used for further improvement of the performance of the car detection system. In aerial images there may be details in roofs, windows, etc. of buildings that look like cars and may therefore lead to false detections. These false detections can be eliminated by using the results from other processing stages of the intepretation of the digital aerial images [42], [16].

In this work, street layer, which contains road information, can be obtained from land use classification process [42] for further improvement of our system. The approach for land use classification applies spectral classification techniques to multi spectral digital aerial images with RGB and NIR (near infrared) channels. A support vector machine is trained to perform an initial classification based on these RGB and NIR images. Additionally the height data generated after aerial triangulation and dense matching are used in a second classification step. The classification results are data layers for streets, buildings, trees, low vegetation or water (for details see [42]). In the context of car detection we are interested in the street layer only. The street layer can be used as a site model to mask the possible regions such as road or parking lots, where cars may be located. This helps to reduce the number of false positives.

III. Experiment and results

The aim of our experiment is to demonstrate the robustness of our framework for car detection from aerial images.

A. Data sets

In this paper we use two different datasets. Both datasets were acquired by the UltraCamD camera from Vexcel wholly owned by Microsoft Corporation. The high resolution panchromatic images - used for car detection, aerial triangulation and dense matching - have a size of 11500 pixel across-track and 7500 pixel along-track. The multi spectral low resolution images - used in the initial step of land use classification - have a size of 3680×2400 pixel. The first dataset was acquired in the summer of 2005 from the city center of Graz, Austria. It consists of 155 images flown in 5 strips. The along-track overlap of this data set is 80%, the across-track overlap is approximately 60%. The ground sampling distance is approximately 8 cm. Therefore, a car is supposed to capture approximatly 24×50 image pixels. The second dataset was acquired in the winter of 2005 from the city center of Philadelphia with the UltraCamD. It consists of 158 images with an along-track overlap of this data set is 90% and an across-track overlap of approximately 60%. The ground sampling distance is approximately 10 cm. The high overlap was chosen to allow automatic aerial triangulation and dense matching.

In this paper, only four typical subimages are taken from these huge original data sets to form training and test sets. Among them two images are from Graz city, the other two are from the images of Philadelphia city. Each subimage has a size of 4000×4000 . We use gray values of those images for training and testing our framework. From now, we will refer to the training set and test set to the sets of these subimages, namely *Graz* and *Philadelphia* datasets. The test sets are separated from the training sets. Each test set contains 324 and 1495 cars, respectively.

B. Training

Depending on the resolution of the aerial images, we can choose the scale for image samples with a reasonable size. Because we know the resolution of the original aerial images, we can specify the rectangle for the subimage patches of cars. The size of the patch has to be carefully chosen to cover the area, which contains a car in the middle and four small surrounding bands (see Figure 2(a)). This is done in order to include some context information of a car so that the car is considered together with its surrounding background. Usually the boundary of a car is a rectangle with the length doubles its width. In our case, we have chosen the patch size to be 35×70 pixel. The patch size is set once and fixed before the training process begins.

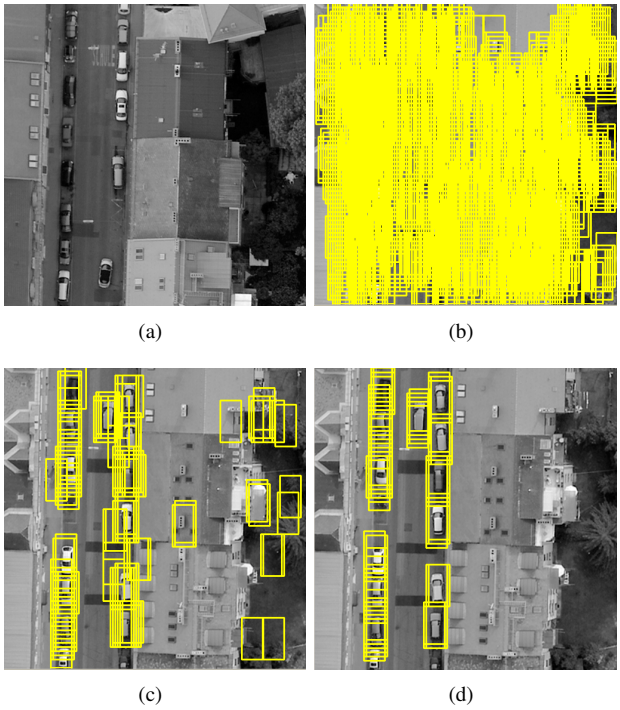
We start with a random classifier. The classifier is improved on-line after labeling training samples by the user. Thus, we make use of the advantages of active learning. During the training process we have labeled 1420 training samples. There are 410 positive samples, each sample contains a car, and 1010 negative samples, each contains diverse background image patches (for a few examples see Figure 2). This whole interactive training process takes approximately 4 hours[1]. The more informative the samples are the faster the system can learn. This is true since our system is trained by an on-line mechanism. Moreover, the training samples can be diversified and adjusted during training process to capture the variances of the real data. The number of positive samples is much less than the number of negative samples we need to train the

[1]Since we are training on-line a classifier is available at each time. Thus with fewer training examples an acceptable results can be obtained after about 2 hours. The longer the training (i.e. the more samples are labeled) the better the performance will be

(a) (b)

Fig. 2. Example of positive (a) and negative (b) labeled training samples during the on-line training process.

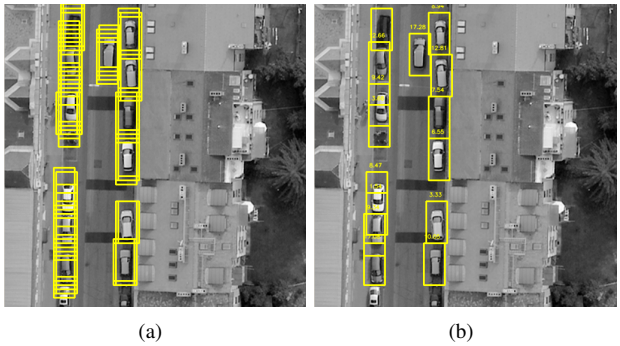
system, which comes from the fact that the variability of the background is much larger than the one of cars. In comparison with other object (car) detection systems, our system needs quite small number of image samples for training. As we can see in Figure 3, after several iterations almost all cars which have rather clear appearance and fit to the (angle of the) detector are detected. Figure 4 depictes the detected objetcs without refining step (a) and the detected points after applying mean shift clustering (b). Figure 5 shows the continuous improvement of the training classifier over time (i.e. number of labeled training samples).



(a) (b)

(c) (d)

Fig. 3. Learning process: Improvement of classifier performance - (a) original subimage, (b) result after training the classifier with only one positive sample, (c) after training with 10 samples and (d) the final result without post processing after training with 50 samples.



(a) (b)

Fig. 4. Postprocessing: (a) Raw output of the classifier applied on a subimage, (b) after combing multiple detections by mean shift based clustering.

C. Performance evaluation

We show the results of our framework for the two data sets: Graz dataset and Philadelphia dataset, each of which contains

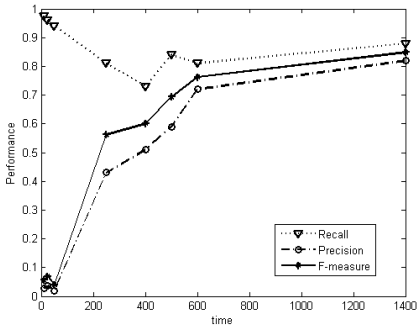


Fig. 5. Learning curves versus number of training examples.

324 and 1495 cars, respectively. Figure 6 and Figure 8 shows the result of car detection in several subimages. The subimages show complicated backgrounds of urban scenes where cars appear as small objects and there are many car-look-like objects. The cars also appear in slightly different view angles, different contrast, lighting condition, etc. Many cars are severe occluded by buildings or trees, or over dominated by the shadow of itself, or have very low contrast. As one can see, all the cars with good features have been detected, even almost all difficult ones could also be detected. For some cars which are partly occluded, they might be detected, might be missed. For some objects that look like cars, they might be reported as cars but with low confidence value and have been removed at post processing stage. The system also works well in dealing with slightly different size of cars. We have trained our system on samples of subimage patches of cars with a size of 35×70 and applied for both datasets of Graz and Philadelphia with ground sampling distance is approximately 8 cm and 10 cm, respectively.

For a quantitative evaluation, we use the common measurement for object detection problem named recall-precision curve (RPC) [2], with:

$$PR = \frac{\#TP}{\#TP + \#FP} \quad (10)$$

$$RR = \frac{\#TP}{\#TP + \#FN} \quad (11)$$

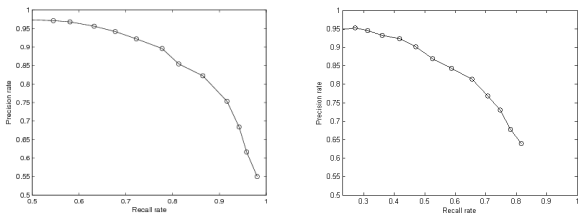
$$Fm = \frac{2 \cdot RR \cdot PR}{RR + PR} \quad (12)$$

(*TP* - true positives, *FP* - false positives, *FN* - false negatives)

The precision rate (*PR*) shows how accurate we are at predicting the positive class. The recall rate (*RR*) shows how many of the total positive we are able to identify. The F-measure (*Fm*) is the harmonic mean that can be considered as trade-off between recall and precision.

For detection, there is always a trade-off between detection rate and false alarm. The RPCs characterize the performance of our framework on the two datasets are given in Figure 7. It depicts the RPC curves of the Graz dataset and Philadelphia dataset with the same parameters setting of the system. For a comparison of the two datasets, we can see that the detector has better performance on the Graz dataset regarding both

Fig. 6. Experimental results of car detection in large aerial images (only subimages are shown). Cars appear with different orientations and maybe occluded in a highly complicated background. The red squares are output of detector, yellow points represent correctly detected cars.



(a) (b)

Fig. 7. RPC curve of the system on *Graz* data set (a) and on *Philadelphia* data set (b).

precision and recall rates. This is the consequence of the differences in the the quality of the acquired images such as the cleanness, the contrast, etc.

In some applications such as the estimation of traffic flow, the coarse information of the site can be given, e.g. road information. In our case, the street layer from land use classification process is used for road verification. This road information improves the detection performance of the system by eliminating false positives. Figure 8 shows the false detection elimination by using road mask. For a fair comparison, beside the regular RPC curves, the RPC curves which take into account site information are also given in Figure 9. As we expected the performance of the system is improved.

Due to the lack of public available datasets for the evaluation of the system and different methods have been employed for evaluation in related works, a fair comparison would be difficult. Although a complete comparison is not met, we claim that our experimental results show that, in general the performance of our framework is superior in terms of the detection rate, the robustness, and especially the efficiency

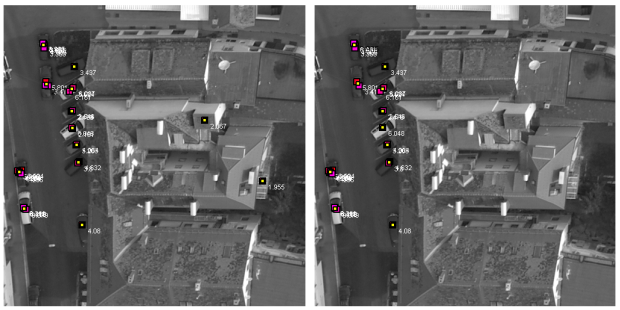
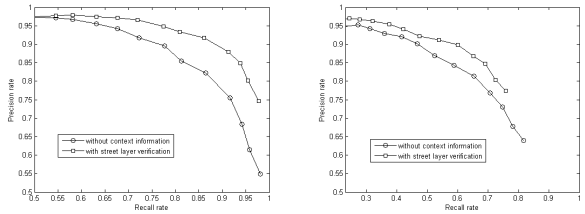


Fig. 8. False elimination: Objects on the roof which have been reported as cars (left image) are removed by using road mask (right image). The red squares are output of detector, yellow points represent correctly detected cars.



(a) (b)

Fig. 9. Increasing detection performance on the *Graz* (a) and *Philadelphia* (b) datasets when including context information (street layer classification).

of automatic process on large scale aerial images [44], [12], [13], [32], [41]. For the detection rate, for instance in [13], result was reported on test data set which contains only 119 cars with the completeness about 80%. Even some related works did not provide clearly their performance evaluation, only some intuitive results were shown [35], [12]. There was no report of related work on the performance evaluation of their system on such large datasets, which have been acquired in different imaging conditions (which are summer and winter of different cities), different ground sampling distance, and different imaging quality.

Our system is applicable for applications such as the estimation of traffic flow, road verification to complete land use classification or recovering surface texture for 3D map generation from digital aerial images. On the other hand, if the prior knowledge of the original image, such as context information or road mask, is given the performance of the system can be certainly improved .

** Exploiting the redundancy in the digital images*

The high overlap of the UltraCamD images results in high redundancy which can be exploited to improve the car detection on no additional costs. A quote from [15]:

"The individual image trigger does not add any cost to image acquisition. One can produce as many images within a flight line as one wishes, with no added costs, and thus increase the traditional forward overlap from 60% to 80% or 90%. One might even consider increasing the side-lap as well, since the only cost increase would be for the additional flight lines, not, however, for the images themselves."

Given the ability of imaging, the high overlaps produce multiple images for each ground point. This can be exploited

Fig. 10. The utilizing of multiple overlaps images with different viewing angles: objects (cars) that are occluded or invisible in one image (left images) can be visible in other image and therefore can be detected (right images)

to get reduced occlusions due to buildings and vegetation for objects (cars). By using this availability of redundant data, the performance of our system can be further improved in a natural way as more data is used. This integration of redundancy provides superior results for certain applications. As one can see in Figure 10, cars which are severe occluded by buildings or trees in one image become visible in other image taken on the same flight. These cars are missed detection in one image but get detected in the other one. Some cars are totally invisible in one image but appear clearly in the other one and will for sure be detected, e.g two cars on the upper left of the subimages on the right. One can think of any combination of these detection results for further improvement of the proposed system or to integrate with other applications. This visual inspection significantly improves the performance of the car detection system certainly with no additional imaging costs. Therefore, for application such as estimation of transportation flow or terrestrial texture restoration, the use of redundancy of available data certainly makes sense. Since the establishment of ground truth is a tedious work for overlapping images, we have not yet statistically evaluated the improvement of using redundancy.

IV. Conclusion and Future Work

We have developed an efficient framework for automatic car detection from aerial images. This is the first proposal to use state-of-the-art machine learning technique, Adaboost, for the detection of cars from large scale aerial images. We used integral image for efficient representation and computation of car features. Three types of features, Haar-like Viola-Jones, orientation histogram and local binary pattern, have been used for generating hypothesis for training the detector. Moreover, a novel on-line version of boosting is used for efficient and robust training of the developed system.

The system functions well in dealing with variances of car appearances in complicated backgrounds of urban aerial images. Experimental results show the superiority and the applicability of our framework for applications including estimation of transportation flow, road verification for completing land use classification or help for restoring texture to complete 3D map generation from digital aerial images.

In principle, the framework can be trained and to detect any object, which is suitable with the feature description.

For future work, the system can be improved and extended with the following aspects:

- Including more data samples for training and diversifying features for car's representation, which would result in improving the generalization of the detector and better performance.
- Use information from aerial triangulation or dense matching. Detect cars in multiple, overlapping images that differ in their viewing angle and automatically combine the results, which yields higher performance for the system.

Acknowledgment

This work has been sponsored in part by the Austrian Federal Ministry of Transport, Innovation and Technology under P-Nr. I2-2-26p VITUS2, the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04, the EU FP6-507752 NoE MUSCLE IST project, and the OeAD's scholarship under the project EZA-894. Parts of this work have been done in the VRVis research center, Graz, Austria (http://www.vrvis.at), which is partly funded by the Austrian government research program Kplus.

References

[1] Y. Abramson and Y. Freund. SEmi-automatic VIsual LEarning (SEVILLE): Tutorial on active learning for visual object recognition. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, 2005.

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.

[3] R. Alba-Flores. Evaluation of the use of high-resolution satellite imagery in transportation applications. Technical report, Intelligent transportation system institute, 2005.

[4] E. Bernstein and Y. Amit. Part-based statistical models for object classification and detection. In *Proceedings of Computer Vision and Pattern Recognition*, volume 2, pages 734–740, 2005.

[5] S. M. Bileschi, B. Leung, and R. M. Rifkin. Towards component-based car detection. In *ECCV Workshop on Statistical Learning and Computer Vision*, 2004.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[7] A. Demiriz, K.R. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[8] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[9] H. Grabner, C. Beleznai, and H. Bischof. Improving adaboost detection rate by wobble and mean shift. In *Proceedings Computer Vision Winter Workshop*, pages 23–32, 2005.

[10] H. Grabner and H. Bischof. On-line boosting and vision. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, volume 1, pages 260–267, 2006.

[11] B. Heisele, I. Riskov, and C. Morgenstern. Components for object detection and identification. In *Proceedings Sicily Workshop on Object Recognition. To appear*, 2006.

[12] S. Hinz. Detection and counting of cars in aerial images. In *International Conference on Image Processing*, volume 3, pages 997–1000, 2003.

[13] S. Hinz and U.Stilla. Car detection in aerial thermal images by local and global evidence accumulation. *Pattern Recognition Letters (to appear)*, 2006.

[14] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, pages 695–700, 2005.

[15] F. Leberl and Szabo J. Novel totally digital potogrammetric workflow. In *Semana Geomatica, IGAC-Bogota, Colombia*, 2005.

[16] F. Leberl, Gruber M., Ponticelli M., Bernoegger S., and Perko R. The ultracam large format aerial digital camera system. In *Proceedings of the ASPRS Annual Convention, Anchorage USA, 5-9 May*, 2003.

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

[18] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proceedings International Conference on Computer Vision*, volume 2, pages 626–633, 2003.

[19] R. Lienhart and J. Maydt. An extended set of haar-like features for object detection. In *Proceedings International Conference on Image Processing*, pages 900–903, 2002.

[20] A. L. C. Marczack, M. J. Johnson, and C. H. Messom. Real-time computation of haar-like features at generic angles for detection algorithms. Research Letters in the Information and Mathematical Sciences - ISSN 1175-2777, Vol. 9, 2005.

[21] H. Moon, R. Chellappa, and A. Rosenfeld. Performance analysis of a simple vehicle detection algorithm. *Elsevier Science*, 20:1–13, 2002.

[22] V. Nair and J.J. Clark. An unsupervised, online learning framework for moving object detection. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, volume 2, pages 317–324, 2004.

[23] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[24] N. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[25] N. Oza and S. Russell. Online bagging and boosting. In *Proceedings Artificial Intelligence and Statistics*, pages 105–112, 2001.

[26] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38:15 – 33, 2000.

[27] Jin-Hyun Park and Young-Kiu Choi. On-line learning for active pattern recognition. *IEEE Signal Processing Letters*, 3(11):301–303, 1996.

[28] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, volume 1, pages 829–836, 2005.

[29] A. N. Rajagopalan, P. Burlina, and R. Chellappa. Higher order statistical learning for vehicle detection in images. In *International Conference on Computer Vision*, volume 2, pages 1204–1209, Washington, DC, USA, 1999. IEEE Computer Society.

[30] P.M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proceedings Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 223–230, 2005.

[31] C. Rudin, I. Daubechies, and R.E. Schapire. The dynamics of adaboost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

[32] R. Ruskone, L. Guigues, S. Airault, and O. Jamet. Vehicle detection on aerial images: A structural approach. In *International Conference on Pattern Recognition*, volume 3, pages 900–904, 1996.

[33] R. Schapire. The boosting approach to machine learning: An overview. In *Proceedings MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[34] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings International Conference on Machine Learning*, pages 322–330, 1997.

[35] C. Schlosser, J. Reitberger, and S. Hinz. Automatic car detection in high resolution urban scenes based on an adaptive 3d - model. In *2nd GRSSilSPRS Joint Workshop on "Data Fusion and Remote Sensing over Urban Areas"*, 2003.

[36] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings Conference on Computer Vision and Pattern Recognition*, volume 1, pages 746–751, 2000.

[37] M. Stojmenovic. Real time machine learning based car detection in images with fast training. *Machine Vision and Applications*, 17:163 – 172, 2006.

[38] K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, pages 228–235, 2000.

[39] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conferecen Computer Vision and Pattern Recognition*, volume I, pages 511–518, 2001.

[40] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *Proceedings International Conference on Automatic Face and Gesture Recognition*, pages 79–84, 2004.

[41] J. Yao and Z. Zhang. Semi-supervised learning based object detection in aerial imagery. In *Proceedings of the Computer Vision and Pattern Recognition*, volume 1, pages 1011 – 1016, 2005.

[42] L. Zebedin, A. Klaus, B. Gruber-Geymayer, and K. Karnera. Towards 3d map generation from digital aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:413–427, 2006.

[43] H. Zhang, W. Jia, X. He, and Q. Wu. Learning-based license plate detection using global and local features. In *Proceedings International Conference on Pattern Recognition*, page to appear, 2006.

[44] T. Zhao and R. Nevatia. Car detection in low resolution aerial image. In *International Conference on Computer Vision*, 2001.