288

Recognizing Cars in Aerial Imagery to Improve Orthophotos

Franz Leberl, Horst Bischof, Helmut Grabner, Stefan Kluckner
Institute for Computer Graphics and Vision
Graz University of Technology
Inffeld. g. 16/II, A-8010 Graz, Austria
{leberl,bischof,hgrabner,kluckner}@icg.tugraz.at

ABSTRACT

The automatic creation of 3D models of urban spaces has become a very active field of research. This has been inspired by recent applications in the location-awareness on the Internet, as demonstrated in maps.live.com and similar websites. The level of automation in creating 3D city models has increased considerably, and has benefited from an increase in the redundancy of the source imagery, namely digital aerial photography. In this paper we argue that the next big step forward is to replace photographic texture by an interpretation of what the texture describes, and to achieve this fully automatically. One calls the result "semantic knowledge". For example we want to know that a certain part of the image is a car, a person, a building, a tree, a shrub, a window, a door, instead of just a collection of 3D points or triangles with a superimposed photographic texture. We investigate object recognition methods to make this next big step. We demonstrate an early result of using the on-line variant of a Boosting algorithm to indeed detect cars in aerial digital imagery to a satisfactory and useful level of completeness. And we show that we can use this semantic knowledge to produce improved orthophotos. We expect that also the 3D models will be improved by the knowledge of cars.

Categories and Subject Descriptors

I.4.8 [**Computing Methodologies**]: Image Processing and Computer VisionScene Analysis[Object Recognition]

General Terms

3D City Models, Semantic Information

Keywords

Aerial Images, Boosting, Car Detection, Object Recognition, Online Learning

1. INTRODUCTION

The systematic creation of models of the real world to support the locational awareness on the Internet, or to grow the current 2D approach to car navigation into a full 3-D experience, is an expensive proposition if based on manual methods. This motivates the work to replace such massive labor by automated procedures. Systems that are currently being set up to produce 3D urban models for Internet applications can rely on a rich literature, a great variety of methods and academic projects that demonstrate fully automatic approaches to 3D reconstruction by shape-from-stereo or laser scanning, e.g. [26].

In recent years our team has inspired the development of a fully automated work-flow to recover 3D city models from highly overlapping aerial images produced by the *UltraCam* from *Microsoft Photogrammetry* (formerly Vexcel Imaging). In its most recent incarnation, each UltraCam-image resolves $14,430 \times 9,420$ pixels. Success in the robust automatic processing of the data depends on a high inter-image redundancy based on an 80% along-track overlap, thus within an individual flight line, and a 60% across-track, thus from one flight line to the next. In much of our work, we employ a ground sampling distance (GSD) for the digital aerial images at approximately 8 cm. At this GSD a typical car gets imaged onto 35×70 pixels.

Intermediate data sets get computed consisting of a Digital Surface Model (DSM) of the terrain, which then gets separated into the "Bald Earth" (a Digital Terrain Model DTM) representing the terrain off which the vertical objects get stripped, as well as those vertical objects themselves. The data exist in the form of "point clouds". Associated with the points, or triangles formed from the points, are patches of photo texture. From those intermediate results, one builds both the 2-dimensional orthophotos as well as the 3-dimensional "city models". The orthophotos are created by projecting the photo texture either onto the Bald Earth DTM for so-called "Traditional Ortho Photos" or onto the DSM to produce a so-called "True" or "Reflective Surface Ortho Photo". The 2-dimensional ortho photos provide a simpler data structure, smaller data volume and a great ease of use and orientation. By contrast, the 3-dimensional models consist of more data, need a more complex data structure and are more difficult to use and navigate. Therefore all location-aware web sites offer imagery in the form of 2D orthophotos.

In this paper, we want to go a step further and develop semantic knowledge about the objects in the terrain, and initially use that knowledge to affect the orthophotos. In subsequent steps, we will use that knowledge also to improve the 3-dimensional city models.

The paper is structured as follows. We first review the 3D city modeling approach into which the semantic knowledge needs to be fed. Then we address object recognition in aerial images, and in subsection 3.2 we introduce an on-line Boosting variant. In section 4 we demonstrate an initial application of this semantic knowledge creation to detect cars. We also demonstrate how the removal of cars from an aerial image improves the 2-dimensional visualization in the form of an orthophoto. We will finally conclude that the method produces reliable and accurate results that outperform a manual approach of car detection and removal.

2. CURRENT 3D CITY MODEL GENERATION

2.1 Camera Station Information

The current approach of city modeling starts with a block of overlapping aerial photographs, typically obtained from a series of parallel flight lines, each representing a so-called "strip" of overlapping images. The coordinates and attitude of each camera station (thus the exterior orientation) are computed by an automatic process that searches for a great number of tie-points among overlapping images, perhaps 10,000 in a single image, and then computes the exterior orientation from the geometry of the image block. This relies on "redundancy" not only by image overlap, but also by the use of a very high number of tie points. The approach is described in [28].

2.2 Surface Modeling

The computed camera orientation parameters feed into an area based image matching algorithm to produce a dense range patch for each of the input images. Those patches then get converted into a seamless digital surface model DSM. The particular implementation of the approach has been described by [11]. The range images are computed from three input images (a reference image and its two immediate neighbors) using a plane sweeping approach. The plane sweep uses the normalized cross correlation as similarity measure and produces a so-called 3D depth space which contains the depth hypotheses and their associated correlation values. The final range image is computed using a semi-global optimization approach proposed by [11]. It employs a redundant and overlapping set of patches, each derived from an image triplet. Given an 80% forward-overlap, each ground point will be imaged onto 5 images. This overlap results in three image triplets per ground point, ready to get merged.

2.3 Creating Data Products in 2D and 3D

Combining the resulting terrain surface with the image textures produces a result as shown in Figure 1. The actual work-flow is fairly complex and includes a separation of the surface data into the Bald Earth and its vertical objects, and the point clouds do get triangulated and thinned out to be acceptable in an Internet application. At this time, the methods have advanced to the point where data can get produced fully automatically, and with minimal need for manual edits.

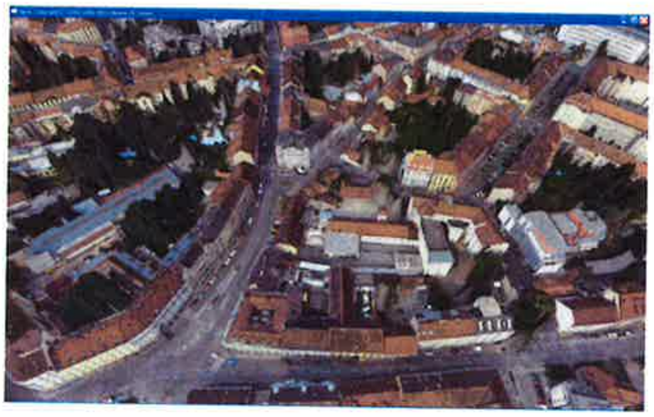


Figure 1: Automatically generated 3D model from a part of Graz. The entire model encompasses 493098 triangles. The image block consists of 155 aerial photographs taken with the *UltraCam$_D$* camera. Courtesy Microsoft-Virtual Earth, K. Karner.

2.4 Adding Semantic Knowledge

The drawback of these automatic reconstruction methods is that only point-clouds (converted into triangles) and textured 3D models are provided which do not present any semantic information. Therefore, we cannot query the data sets according to content nor can we produce specific thematic maps of specific object classes. In order to have a proper interpretation of the scene and to build better 3D models higher level knowledge about the object is required. This has been recognized by recent research work, for example [5, 8]. We believe that semantic information can be extracted fairly reliably, and that this increases the usefulness and value of the data tremendously.

Let us therefore proceed with a presentation of a promising method for object detection applicable to aerial images, and show how this affects the 2-dimensional orthophotos., using the example of cars. Not only do we need to recognize cars, but we also need to remove them from the image and replace the image texture by inpainted texture.

3. OBJECT DETECTION

3.1 Background

In terrain images, we have permanent objects that we want to represent in a location-aware system. However, we also have objects that are irrelevant because they are not a permanent feature of the terrain. This includes cars and people. We initially focus on cars, although considerable work has been done already to detect faces [22, 25], or pedestrians [17], cars [1], bikes [15], and other visual objects. One sometimes denotes this as a "visual categorization" as opposed to "object recognition" [7, 15]. See [19] for a recent overview of research in the area of visual categorization.

The predominant paradigm for object detection is now based on classification approaches which scan the whole image by sliding a window over it, at different resolutions, to extract features such as edges, corners, texture, and classify this window as one containing the object of interest or not. Usually some post-processing by a local maxima search or sim-

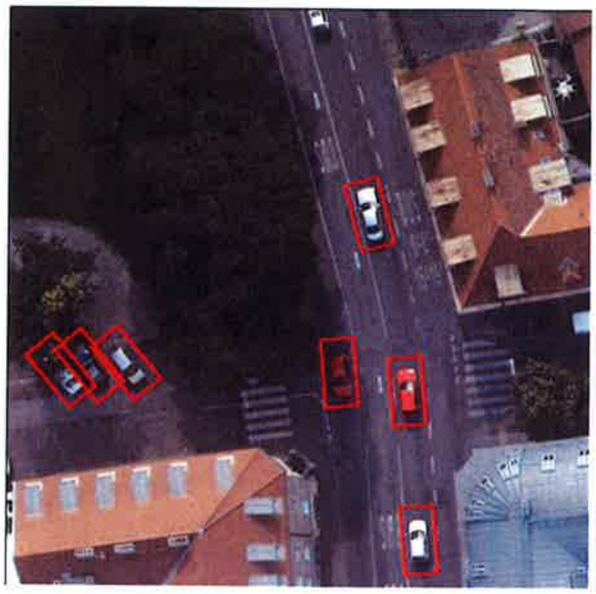


Figure 2: A window of 600 × 600 pixels taken from a test area consisting of 4500 × 4500 pixels. The red rectangles represent the manually developed ground truth.

ilar approaches is necessary to avoid multiple detections. At the core of these object detection approaches is a classifier, e.g., AdaBoost [9], Winnow [13], neural network [22] or support vector machine [24]. The proposed approaches have achieved considerable success in the above mentioned applications. We present in this paper an approach that follows this paradigm; as classifier we use an on-line variant of Boosting [10] to be described in the next section.

3.2 Boosting

Boosting is a classifier combination method, which combines several "weak" classifiers to form a "strong" one. Many researchers have analyzed and applied Boosting for different tasks. There are many different variants of Boosting which have been proposed (e.g. Real-Boost [9] and LP-Boost [6]). We use the discrete AdaBoost algorithm, proposed by Freund and Schapire [9].

The algorithm works as follows: We have 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. A weak classifier h^{weak} is trained using $\mathcal{X}$ and $p(\mathbf{x})$. We only require that the weak classifier perform slightly better than random guessing. Therefore, for a binary decision task the error rate of the classifier must be less than 50%. The weak classifier h_n^{weak} then gets a weight $\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-e_n}{e_n}\right)$, where e_n denotes the error of the classifier. Depending on the performance of the weak classifier, the probability $p(\mathbf{x})$ gets updated. For misclassified samples, the corresponding weight gets increased, while for correctly classified samples the weight is decreased. As a consequence the algorithm focuses on examples that are difficult to learn. This process gets iterated. A new weak classifier is added at each Boosting iteration until a certain stopping criterion is met.

From the resulting set of N weak classifiers $h_n^{weak}(\mathbf{x})$, a strong classifier $h^{strong}(\mathbf{x})$ is generated, by a linear combination:

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

$$h^{strong}(\mathbf{x}) = \text{sign}(conf(\mathbf{x})) \qquad (2)$$

$conf(\mathbf{x})$ is bounded by $[-1,1]$, therefore we interpret it as a confidence measure of the strong classifier.

Similarly, we can apply Boosting for feature selection, as introduced by Tieu and Viola [23]. The basic idea is that each feature (together with a threshold that is trained) corresponds to a weak classifier. The application of Boosting to these features gets us an informative feature subset.

The training for feature selection proceeds in a manner similar to the described algorithm. From a set of possible features $\mathcal{F} = \{f_1, ..., f_k\}$, and at iteration n the best feature forms a weak classifier h_n^{weak} which corresponds to the selected feature f_n. The weights of the training samples are updated with respect to the error of the chosen classifier.

3.2.1 On-line Boosting for Feature Selection

Feature selection via Boosting needs all training samples to be available in advance. We use an on-line feature selection algorithm [10], based on an on-line version of AdaBoost [16]. This requires to each boosting step to be performed on-line.

The basic idea of on-line Boosting is to estimate the difficulty of a sample by propagating it through the set of weak classifiers. This can be interpreted as modeling the information gain with respect to the first n classifiers and code it by an importance factor λ for doing the update of the $n+1$-th weak classifier. As proven in [16] the result of the classifier using on-line Boosting converges statistically to the one obtained by off-line Boosting as the number of iterations $N \to \infty$. When presented the same training set multiple times, on-line and off-line Boosting achieve the same results.

For selecting features by on-line Boosting, "selectors" are introduced. On-line Boosting is then not directly performed on the weak classifiers, but on the selectors. For that purpose, a selector $h^{sel}(\mathbf{x})$ consists of a set of M weak classifiers $\{h_1^{weak}(\mathbf{x}), \ldots, h_M^{weak}(\mathbf{x})\}$ and selects the one with minimal error.

$$h^{sel}(\mathbf{x}) = \arg\min_m e\left(h_m^{weak}(\mathbf{x})\right) \qquad (3)$$

When training a selector, its M weak classifiers are trained and the one with the lowest estimated error is selected. Therefore, a selector can also be seen as a classifier which switches between the weak classifiers. As in the off-line case, each weak classifier corresponds to a single feature, i.e. the hypothesis generated by the weak classifier is based on the response of the feature.

The on-line AdaBoost training algorithm used for feature selection proceeds as follows: A fixed number of N selectors $h_1^{sel}, .., h_N^{sel}$ is initialized with random features. The selectors and the corresponding voting weights α_n are updated, as soon as a new training sample $\langle \mathbf{x}, y \rangle$ is available. The weak classifier with the smallest estimated error is selected in each selector. For updating the weak classifier one can use any on-line learning algorithm. Finally, the weight λ_n of the sample is updated and passed to the next selector h_{n+1}^{sel}. The weight is increased if the sample is misclassified by the current selector or decreased otherwise. A linear combination of the N selectors gives the strong classifier:

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

For more details see [10].

3.2.2 Image Features

Using features instead of raw pixel values increases the robustness and integrates invariance in the classifier. In principle we can choose among many different features, but since we are working with large images, and since these large images need to be scanned by the classifier we look for features that can be computed quickly, for example by means of an integral image [25]. Our choices are Haar-like features [25], orientation histograms [12, 20] and a simplified version of local binary patterns [14].

3.2.3 Object Recognition versus Classical "Classification"

Classification of pixels or pixel groups is part of a standard 3D city modeling system [28]. The image content gets separated into regions that typically describe building roofs, circulation spaces such as streets and parking areas or driveways, vegetation, grass surfaces and water. Classification results feed into the separation of vertical objects from the Bald Earth, to name an example.

However, "object recognition" differs from "classification" in its goal. The goal of object recognition is to describe and detect an object as a whole, taking into account its texture and shape. Classification is pixel oriented, a single pixel or a pixel and its neighborhood are the input and a class label for a single pixel is the output. Nevertheless these processes may support each other as we discuss later.

4. CAR DETECTION AND REMOVAL

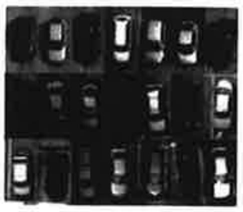
4.1 Training

The most important element for success in car detection is to properly train the classifier. This classification problem is treated as a binary problem, car versus background. Usually this requires a large amount of pre-labeled data, perhaps in the order of ten-thousand images. However since we have an on-line learning method which is sufficiently fast for interactive work, we attack training as an interactive learning problem. The key idea is that the user has to label only those examples that are not correctly classified by the current classifier. In fact, it has been shown by the active learning community [18] that it is more effective (increase the learning speed, or minimize the number of labeled samples) to sample at the current estimate of the decision boundary than at the unknown true boundary. This is exactly the approach we are following.

We evaluate the current classifier on an image. The human supervisor labels additionally "informative" samples, e.g. marks a wrongly labeled example which can either be a false detection or a missed car. The new updated classifier gets applied again on the same image or on a new image, and the process continues iteratively until a satisfactory detection performance is achieved. This is a fully supervised interactive learning process. This process permits us to update the parameters of the classifier in a greedy manner with respect to minimizing the detection error. This results in very fast training with a minimum number of samples (see experimental results). It avoids labeling redundant samples that do not contribute to the current decision boundary.

4.2 Detection

After training, detection results from the exhaustive application of the trained classifier on the images. Since we know the resolution of the image we do not need to search cars at various scales. However, we need to cope with the car's orientation and therefore do search cars at different image rotations. Instead of training the classifier with different orientations we train it at one canonical orientation, and evaluate it by rotating the image with 15 degree increments. An option exists to have the detector be rotated by computing the features at different angles for the detection process. The most efficient approach is the use of features that can be rotated quickly. 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 will occur[1]. Using just a simple threshold results in many overlapping detections. Therefore, post processing refines and combines these outputs.


(a)

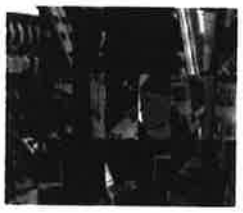

(b)

Figure 3: Some positive (a) and negative (b) examples that have been labeled by the human during the training process of the classifier.

4.3 Post Processing

For post processing we could use non-maximum suppression. A slight improvement of the localization can get achieved using the mean-shift algorithm [4]. The confidence values of the classifier are an input to the mean shift procedure. The mean shift algorithm is non-parametric and iteratively

[1]For car removal we can accept some false detections because in most cases the inpainting procedure will not change the image anyway in a significant manner

locates density extrema or modes of a given distribution. We rely on a Gaussian kernel, its width is related to the size of the car. Starting from an initial location the local mean shift vector maximizes the underlying probability density function in a gradient descent manner.

For additional improvements of the detection result, we like to use additional context information such as the street layer from a standard pixel based classification using support vector machines, as mentioned in section 3.5 and reviewed in [28])

4.4 Removing Cars and Texture In Painting

The detected cars need to get removed from the images. A procedure is needed to avoid visual artifacts where texture is now missing. Inserting plausible values is being achieved by a process called "inpainting". Many different approaches for inpainting have been developed. We use a variant based on minimizing an energy functional referred to as total variation [3]. We obtain satisfactory results with no obvious visual artifacts. More sophisticated inpainting methods could be used also, such as [2] or methods based on graph-cuts.

5. EXPERIMENTAL RESULTS

The experiments serve to assess and demonstrate the efficiency of the on-line training and robustness of our framework for car detection from aerial images. They also demonstrate the successful removal of cars from the images and their replacement by generic texture for use in 2-dimensional orthophotos.

5.1 Accuracy measures

For a quantitative evaluation, we use so-called recall-precision curves (RPC) [1]. We manually establish a reference set of cars, representing the ground truth with $\#nP$ cars. Of the total detected cars, $\#TP$ are the true positives and $\#FP$ the false positives. The precision rate (PR) shows the accuracy of the prediction of the positive class. The recall rate (RR) shows how many of the known total number of positive samples we are able to identify. The F-Measure (Fm) is the harmonic mean between RR and PR , a type of average between recall and precision rate. For a visual evaluation of the detector, we plot RR against $1 - PR$.

$$PR = \frac{\#TP}{\#TP + \#FP} \tag{5}$$

$$RR = \frac{\#TP}{\#nP} \tag{6}$$

$$Fm = \frac{2 \cdot RR \cdot PR}{RR + PR} \tag{7}$$

What is a "correct detection"? We accept a detection as "correct" if and only if the center of the detection corresponds to the annotated ground truth car with a maximum city block distance of approximately 1.8m (representing 22 pixels in the example images). In addition we require that the detected orientation be within 16 degrees of the ground truth. These values remain constant for all experiments.

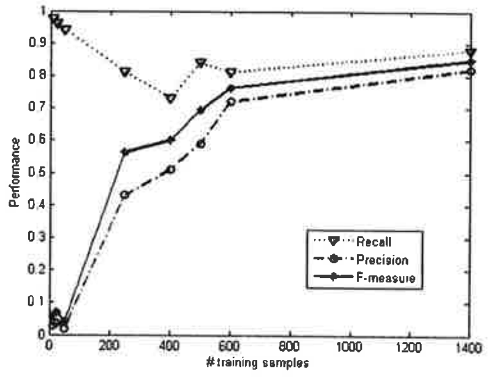


Figure 4: Performance of the trained classifier versus training time.

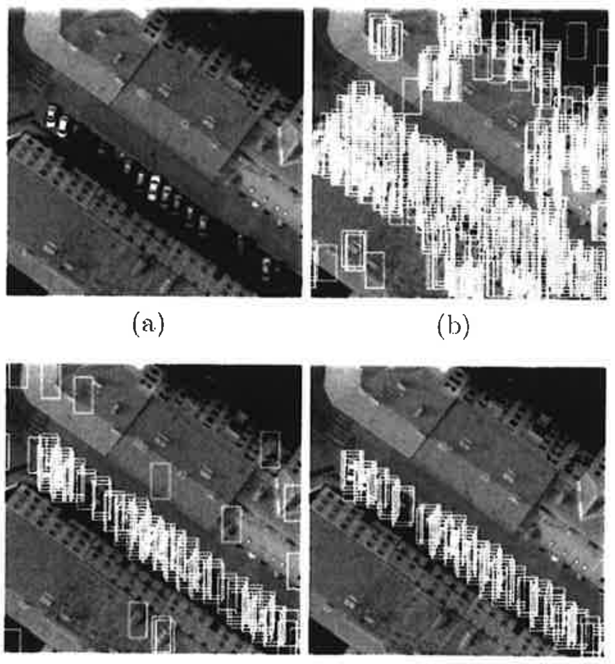


Figure 5: Learning process: Improvement of classifier performance - (a) original image, (b) result after training with only one positive sample, (c) after training with 10 samples and (d) result (without post processing) after training with 50 samples.

5.2 Data Set

Our initial experience is with aerial photography of the Graz city center (Austria). The images were acquired by the *UltraCam$_D$* digital aerial camera developed by *Microsoft Photogrammetry*. This camera produces 4 color channels in red-green-blue-near infrared, and the images used initially are at a ground resolution of 8 cm. The radiometry is presented with 16 bit per color channel, with a verified range between 12 and 13 bit. For all the experiments reported color has not been used. The images are converted to gray-scale and only the gray scale information is processed.

The fixed ground sample distance GSD of the aerial images supports a fixed-size rectangle to reflect the size of a typical passenger car. That rectangle is to cover the car in in its center and some surrounding texture. This is to include some context information of a car so that the car is analyzed with its surrounding background. Usually the boundary of a car is a rectangle with a length twice its width. In our case, we have chosen the patch size to be 2.8 meters $\times$ 5.6 meters or 35 $\times$ 70 pixels, respectively. Figure 2 is a typical test image with the annually collected ground truth rectangles overlaid in red.

The initial training and testing sets are selected on two non-overlapping parts of the aerial images.

5.3 Training

We start with a random classifier which consists of 250 selectors, each containing 500 weak classifiers. The classifier is improved on-line by the user. During training we label 1420 samples, of which 410 are positive, each sample containing a car, and 1010 are negative, each showing diverse background patches (for some examples see Figure 3). The more informative the samples are, the faster the system learns. Moreover, the training samples can be diversified and adjusted during training to capture the variability of the real images. In comparison with other object (car) detection systems, our approach needs only a small number of training samples. This is a benefit resulting from the on-line training procedure. Figure 4 depicts the performance values PR, RR and Fm versus the training time (number of samples) of the classifier.

Figure 5 presents the result of the training session. Using 10 to 50 training samples and several training iterations, all cars which have a distinct appearance and fit to the (angle of the) detector are indeed detected. These examples demonstrate also that the performance of the classifier improves dramatically with adding only some ten informative training samples.

5.4 Results

Fig. 6 presents some detection results. We processed images with 4500 $\times$ 4500 pixels, containing 500 cars (only counting cars visible by more than 50%, and excluding vans and trucks). In Figure 6, for illustration we show patches of 400×400 pixels. One can clearly see that of the 9 cars visually identifiable in the patch (a), 9 were detected; in patch (b), of a total 40 visually identifiable cars, 37 were detected.

To make use of the detection result, we have to remove the cars from the images. This is demonstrated in Fig. 6(e) and (f), using an inpainting algorithm that is shown to not generate artifacts. An observer will not be able to see where a car has been removed and will assume that the image was taken of an urban scene without cars.

At this time, we have yet to start taking advantage of the high overlap of *UltraCam$_D$* images. We can at this time only speculate that the use of multiple images covering each car will increase the car detection result, and this will be achieved at no additional cost of training nor of imaging.

Recall that in traditional aerial mapping, imaging overlaps are minimized due to the cost of film and film processing. Therefore those overlaps are optimized to achieve a simple stereo-coverage with each ground point being on only two images. The lack of any variable costs for digital images suggests that one increase the in-flight forward overlap from 60% to 80% or 90%. This not only will help in any detection scheme, but it will also alleviate the notorious limitations of urban mapping that are caused by occlusions, an inability to look into street canyons, and vegetation hiding relevant objects.

Another result yet to get developed is the use of the point clouds obtained from the 3D modeling work. We have in our initial experiments focused on a demonstration that car detection can be done within useful error margins. Steps to reduce those margins, and to creatively apply the results in the 3D environment, will be taken next.

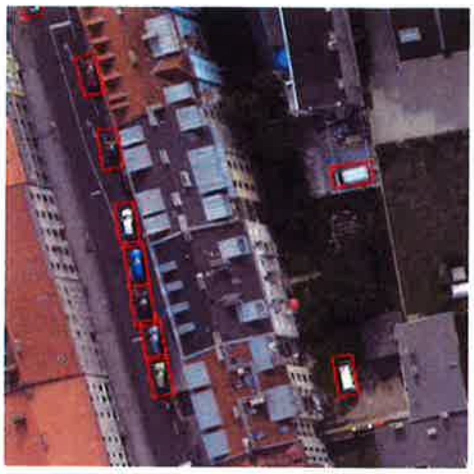
6. CONCLUSIONS

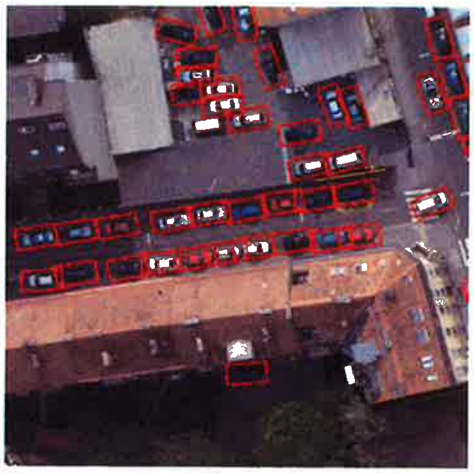
6.1 Semantic Knowledge

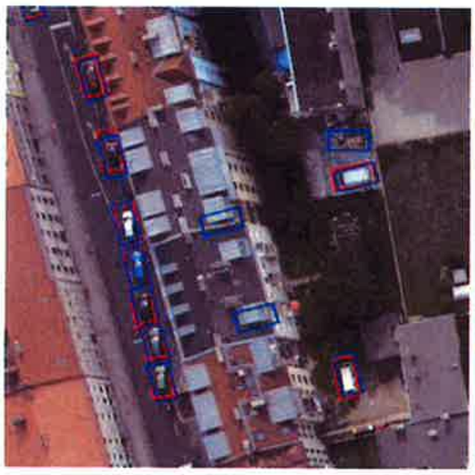
We are making the case for the development of rich semantic knowledge about the objects presented in aerial digital photography. We believe that this is the "next big step" in automatic 3D modeling of the human habitat. The semantic information consists of labels attached to objects in the scene. This is a classical task of object recognition/classification. The paper briefly reviews how this task has been tremendously advanced by the vision research community in recent years. We believe that automatic and robust procedures can be developed for recognizing most common objects in aerial photography, such as people, cars, trees, buildings, windows, doors, chimneys, sky lights etc. We refer to these as "human scale objects". Some of these objects must be maintained in a 3D model of an urban environment, others are of such a transient nature that they are to be removed. And some can be replaced by generic models rather than a specific 3D representation of the specific object, if that object is for example a tree or shrub. Semantic information will have a significant effect on many tasks. We think of the way we search in images, how we can compress, publish and broadcast them on the Internet, how urban scenes get visualized.

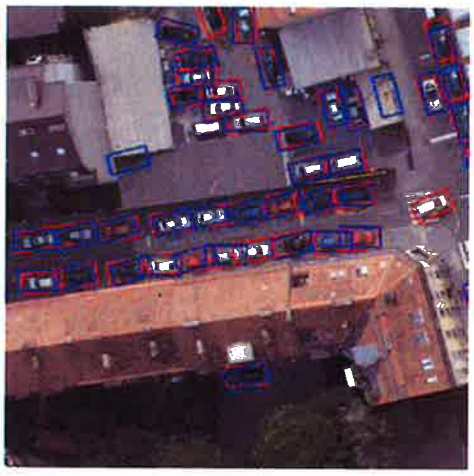
6.2 Detecting Cars

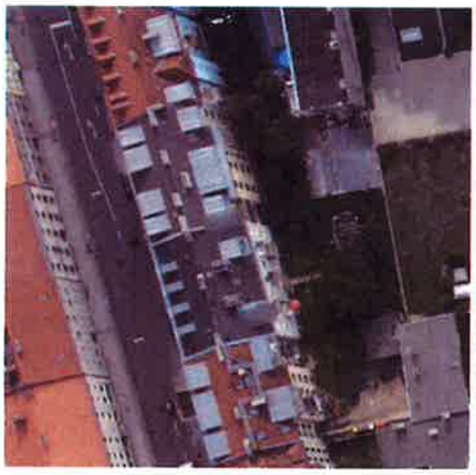
Imagine a city like our town of Graz (Austria) on 200 square kilometers with 250,000 inhabitants and 25,000 buildings. At a GSD of 8 cm and an image overlap of 80% along-track, and with 32 flight lines from 60% across-track overlaps, this will be imaged into a block of 3,000 photographs, representing a raw input data set with 1.3 TB. In such a city and at


(a)


(b)


(c)


(d)


(e)

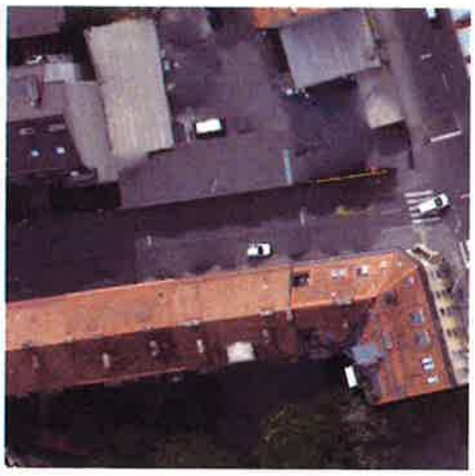

(f)

Figure 6: Sub-figures (a) and (b) show parts of our test images. The ground truth is indicated by a red rectangle around the car. The detection result of our classifier (blue) are compared to the ground truth in (c) and (d). The inpainting results are shown in (e) and (f).

any given time, one might find 80% of all cars parked in or driving on the streets. At 1 car per every two inhabitants, there might be $\approx 100,000$ cars imaged.

Of course, these are 3,000 images because of the high level of redundancy. If one were to just cover the entire city so that each patch on the ground is on just one image, the number would reduce to less than 400.

The car detection task is to scan through the 3,000 aerial images to find the 100,000 cars. To succeed, we are taking a first step of describing a particular instance of an object detection system, apply it to cars and use the result to change published imagery. Cars are an obvious clutter in a 3D city model. Therefore we want to detect and remove them.

Our efficient car detector exploits an on-line Boosting algorithm for training the detector. We use on-line learning with a human to build a good detector from only a few labeled samples, and we can successively improve the detector to achieve satisfaction with the performance. We believe that our procedure can be applied to a range of different objects, not just cars. Of urgent interest are people.

In our experimental work, we demonstrated that we can train the detector with just a few hundred samples. This is far less than other approaches need, for example [27] report results with 7800 (positive and negative) examples. We believe that we can decrease the training effort perhaps to a level where just two samples are needed.

If we scale up our current results, as obtained from only two images with 500 cars, to apply to 400 non-redundant images with 100,000 cars, we would achieve an automated detection of 95000 cars, we would have to cope with 5000 false detections (which do not hurt for inpainting) and we would use about 130 hours of computing (on a single machine). Scaling the work to use 3000 images for the same area and the same number of cars will be possible once we have started to exploit the high overlap imagery.

6.3 Next Steps

Our hope for further improvements is nourished by the availability of overlapping images, typically 10 images per ground patch. We envision an approach that automatically improves the detector in an unsupervised fashion. The image overlaps will help to obtain additionally positive samples, e.g. when a car is detected in one image and not in the other.

Our optimism is further encouraged by recent unpublished work that employs 3D information from the images to automatically label the false negatives and use them for training. We are also exploring co-training strategies by using extracted height field features to construct a shape-based car model in an appearance driven detection process.

We speculate that a combination of approaches will achieve a better performance and also a greater level of generalization towards use for various objects and types of aerial imagery. We have recently proposed an approach called "conservative learning" [21]. There we have demonstrated that by using a combination of generative and discriminative classifiers and a conservative update strategy a person detector can be learned in an unsupervised manner.

7. ACKNOWLEDGMENTS

This work has been supported by the APAFA Project No. 813397, financed by the Austrian Research Promotion Agency (http://www.ffg.at) and the Virtual Earth Academic Research Collaboration funded by Microsoft.

8. REFERENCES

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

[2] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. Technical Report 02(47), UCLA CAM Report, 2002.

[3] T. Chan and J. Shen. Mathematical models for local nontexture inpaintings. *SIAM J. Appl. Math.*, 62(3):1019–1043, 2001.

[4] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *ICCV*, pages 1197–1203, 1999.

[5] N. Cornelis, B. Leibe, K. Cornelis, and L. van Gool. 3d city modeling using cognitive loops. In *Third International Symposium on 3D Data Processing, Visualization and Transmission, 3DPVT*, 2006.

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

[7] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *In Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2003.*, pages 264–271, 2003.

[8] W. Foerstner. Etrims. `http://www.ipb.uni-bonn.de/projects/etrims`, 2006.

[9] Y. Freund and R. 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.

[10] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. of CVPR 2006*, volume I, pages 260–267. IEEE CS, 2006.

[11] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Proc. ICPR 2006*, pages 15–18. IEEE CS, 2006.

[12] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *CVPR*, pages 53–60, 2004.

[13] N. Littlestone. Learning quickly when irrelevant attributes abound. *Machine Learning*, 2:285–318, 1987. Winnow algorithm.

[14] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002.

[15] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Proc. ECCV2004*, volume II, pages 71–84, 2004.

[16] N. Oza and S. Russell. Online bagging and boosting.

In *Proceedings Artificial Intelligence and Statistics*, pages 105–112, 2001.

[17] D. S. P. Viola, M.J. Jones. Detecting pedestrians using patterns of motion and appearance. In *Proceedings of the Ninth IEEE Conference on Computer Vision (ICCV'03)*, volume 2, pages 734–741, 2003. Pedestria dedection using AdaBoost.

[18] J.-H. Park and Y.-K. Choi. On-line learning for active pattern recognition. In *IEEE Signal Processing Letters*, pages 301–303, 1996.

[19] J. Ponce, M. Herbert, C. Schmid, and A. Zisserman, editors. *Toward Category-Level Object Recognition.* Springer, 2006.

[20] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *CVPR*, volume 1, pages 829–836, 2005.

[21] P. Roth, H. Grabner, D. Skocaj, H. Bischof, and A. Leonardis. Conservative visual learning for object detection with minimal hand labeling effort. In W. Kropatsch, R. Sablatning, and A. Hanburry, editors, *Pattern Recognition 27th DAGM Symposium*, volume LNCS 3663, pages 293–300. Spinger, 2005.

[22] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[23] K. Tieu and P. Viola. Boosting image retrieval. In *CVPR*, pages 228–235, 2000.

[24] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition*, pages 511–518. IEEE CS, 2001.

[26] T. Werner and A. Zissermann. New techniques for automated architecture reconstruction from photographs. In *Proceedings of the 7th European Conference on Computer Vision*, volume 2, pages 541–555. Springer, 2002.

[27] B. Wu and R. Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. In *Proc. CVPR 2007*, pages 1180–1187. IEEE CS, 2007.

[28] L. Zebedin, A. Klaus, B. Gruber-Geymayer, and K. Karner. Towards 3d map generation from digital aerial images. *Int. Journal of Photogrammetry and Remote Sensing*, pages 413–427, 2006.