SEMI-SUPERVISED INCREMENTAL LEARNING OF HIERARCHICAL APPEARANCE MODELS

Susanne Wenzel and Wolfgang Förstner

Department of Photogrammetry
Institute of Geodesy and Geo Information, University of Bonn
susanne.wenzel@uni-bonn.de, wf@ipb.uni-bonn.de
http://www.ipb.uni-bonn.de

Commission III/IV

KEY WORDS: Detection, Building, Structure, Interpretation, Classification, Incremental Learning, Recognition

ABSTRACT:

We propose an incremental learning scheme for learning a class hierarchy for objects typically occurring multiple in images. Given one example of an object that appears several times in the image, e.g. is part of a repetitive structure, we propose a method for identifying prototypes using an unsupervised clustering procedure. These prototypes are used for building a hierarchical appearance based model of the envisaged class in a supervised manner. For classification of new instances detected in new images we use linear subspace methods that combine discriminative and reconstructive properties. The used methods are chosen to be capable for an incremental update. We test our approach on facade images with repetitive windows and balconies. We use the learned object models to find new instances in other images, e. g. the neighbouring facade and update already learned models with the new instances.

1 INTRODUCTION

The interest in geo-applications like Google Earth and Microsoft Virtual Earth has increased tremendously lately. So far, most 3D city models are very simple and without semantic information. Furthermore, the creation of such models needs a lot of user interactions. The need for semantic enrichments is seen in different applications of 3D city models of high level of detail. E. g., for accurate navigation to a certain address one would like to know the exact position of the door. Tasks of urban management or catastrophe management need to know the number of stories and the availability of balconies. Architects who plans a new building need to know the types and number of windows in the neighbouring facades. Automatic procedures for deriving such information are of great use. The present paper focuses on simplifying the development tools for interpreting images of man-made scenes, especially of building scenes.

As building parts often show some degree of symmetry, we address the following problem: Given one example of an object and given the prior knowledge that it is repeatedly present in the same image we can learn its object class appearance. This provides a tool to detect other objects of the same type in other images with minimal need of user interaction. Additionally, we build up an object class hierarchy with a minimal amount of user interactions. As we want to increase our knowledge about class appearances with every new image, we want to propose models that are capable of incremental learning.

The paper is structured as follows. First we give an overview over our concept for semi-supervised incremental learning. Sec. 3 refers to related work and Sec. 4 describes our concept in more detail. Experiments are described in Sec. 5. We conclude with Sec. 6.

2 OVERVIEW

Our objective is to develop an incremental learning scheme which requires the least user interaction. To achieve this goal, we need four procedures: (1) a detector for finding new instances (2) a prototype generator for finding new class candidates unsupervised, (3) a classifier for the envisaged classes and finally (4) an GUI for the interaction between the system and the supervisor. The internal representation of the classes needs to be updatable incrementally.

In order to achieve a clear representation we change the order and start bottom up. So the first task of this work is to automatically identify prototypes, i. e. characteristic instances of their classes for supporting autonomous learning. In general, we cannot assume single prototypes to be sufficient for complex object classes like windows, balconies or doors, but we need to assumethat multiple prototypes per class exist, which possibly are structurally related e. g. by a hierarchy. As an example Fig. 1 shows some instances of windows. We can obviously find a hierarchy of prototypes, e. g. rectangular and arclike windows, which further have subclasses depending on the number of crossbars. We propose a method for identifying prototypes using an unsupervised clustering procedure. For this purpose we use a similarity graph, which is built up recursively by detecting similar objects given only one or a few examples, and take its connected components as clusters. Thus, given a single instance of a class we let the system find as many clusters as possible for that superclass. Based on user's judgement they are specified, e. g. as an object of the same superclass but maybe a new subclass or as background. We learn the object class appearance and use their prototypes $\boldsymbol{p}_c$ to automatically detect probable new instances of class $c \in 1 \ldots C$ in new images.

Secondly, for classification we represent identified classes as a kind of Fisher-images, see (Belhumeur et al., 1997) and combine reconstructive (PCA) and discriminative subspace (LDA) methods, see (Fidler, 2006) for classification of new instances. When detecting new instances in new images we incrementally update our class representations and the object hierarchy.

The methods described in this paper are conceived very general. They can be used whenever one deals with objects which appear multiple times in an image, e.g. in repetitive structures and which occur with similar appearance in other images. As special application this paper addresses the detection and classification of win-

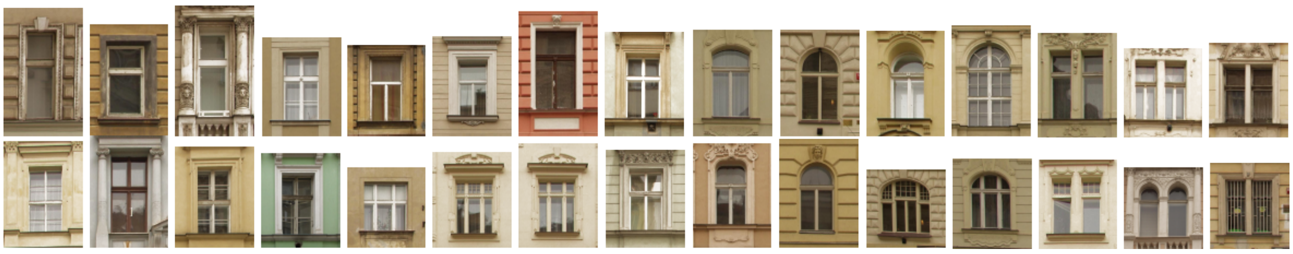


Figure 1: Some examples for appearances of class window.

dows and balconies in single facade images and the incremental learning of their broad appearance over the time.

3 RELATED WORK

Learning from few examples, usually known as one shot-learning is well investigated, cf. e.g. (Fei-Fei et al., 2004) and (Li et al., 2007). Fei-Fei et al. present a method for learning object categories from just a few training examples and obtain the prior knowledge from object categories which were previously learnt. In contrast to their approach, we explicitly use the prior knowledge that the object appears repeatedly to find new instances and learn the class appearance from these instances. Given the knowledge of existence of similar but different object classes we make hypothesis of new object classes or subclasses which might be accepted from the user or not.

The idea of using similarities between neighbouring objects is not totally new. Sivic and Zisserman (2006) aim at finding key persons in a video. They propose to initiate the learning of the appearance model by identifying the person in a single image and use a tracking procedure to generate a large set of training samples. Thus the inherent similarity graph is built up using pairwise similarities and explore the chaining effect to obtain dissimilar instances of the person.

Van Gool et al. (2007) and his group aim at estimating vanishing points in strong perspective facade images. They search for rows and columns of similar image patches. They first establish a similarity graph between interesting image features, from which they derive a similarity chain using a maximum spanning tree (MSP). Consecutive nodes in the MSP are then taken to be image features which are arranged in rows or columns, then allowing to estimate the vanishing points.

Our method of finding prototypes is comparable to that of Sivic and Zisserman (2006): As our goal is to automatically derive a hierarchy for appearance classes, we provide one or several instances of the class to be modelled and let the system find as many prototypes for that class.

4 A CONCEPT FOR INCREMENTAL LEARNING

In this section we want to describe our concept for incremental learning in more detail. We start with the determination of prototypes.

4.1 Getting Prototypes

A prototype is a representative instance of its class. An instance is representative if it may be taken as surrogate for the probability density function (p. d. f.) of the class during classification, as e. g. in case based classification. Therefore one might need several prototypes being representative for a class. For choosing prototypes one could use characteristic points of the features joint p. d. f., like the mode or the mean.

Thus the goal of the following algorithm is to automatically establish a large set $\mathcal{X}$ of instances χ_i of the envisaged super class and by clustering identify appearance subclasses c, which then are the basis to determine a prototype p_c for each cluster.

We assume the class to be representable by the multidimensional p. d. f. of the feature vector describing each instance of the class. As we want to initiate the prototype detection with as few training data as possible we assume the density values of the p. d. f. between neighbouring nodes to be large enough to produce a chaining effect. The chaining effect thus is exploited to minimize the required user guidance. Otherwise, multiple instances need to be given as training samples.

Our method assumes the image to contain multiple instances of the envisaged concept. In addition, we assume to have a similarity measure $s(\chi_i, \chi_j)$ for two different instances, e. g. replacing the probability density $p(\boldsymbol{x}_i - \boldsymbol{x}_j)$ of the difference of the correspondent feature vectors.

For the moment we do not exploit the spatial configuration of the instances.

Finding a large subset $\mathcal{X}$ of $\mathcal{X}^0$ from a given instance χ_0 Starting from an initial instance $\chi_0(\boldsymbol{x}_0)$ with feature vector $\boldsymbol{x}_0$ we search for all instances χ_i in the set $\mathcal{X}^0$ of all candidate instances having a similarity $s(\boldsymbol{x}_0, \boldsymbol{x}_i) > T_1$ better than a given low threshold T_1. Starting from these instances we again search for all instances with a similarity better than T_1. This recursion allows to reach all instances bridged by the chaining effect, but possible also instances not belonging to the envisaged class.

At the moment we represent each instance χ_i by the complete intensity matrix, thus $\boldsymbol{x}_i$ contains all intensity values of χ_i. We work on rectified metric facade images so we need not deal with perspective effects. We choose the normalised cross-correlation coefficient ρ as our similarity measure, as it can deal with intensity changes (e. g. due to shadows) and we need no rotation invariance. So $s(\chi_i, \chi_j) = \rho(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Given an image I and a single example [1] χ_0 of size $a \times b$ we scan the whole image for similar objects. This is realized by the cross-correlation function ρ with $\rho(r, c) = [\rho(\boldsymbol{x}_0, \boldsymbol{x}(r, c))]$ between given template $\boldsymbol{x}_0$ and image patches of size $a \times b$ within the whole image I. Thus we formally take all positions as candidate set $\mathcal{X}^0$. In the first iteration we select all local maxima in ρ with $\rho(i, j) > T_1$, cf. the function non-maximum-suppression in line 1.2 and 1.8 of Alg. 1. The given example χ_0 defines the seed node v_0 of a graph $\mathcal{G}(\mathcal{V}, \mathcal{E}; \mathcal{S})$ and image patches χ_i of size $a \times b$ around positions of found local maxima define the first sequence of additional vertices $v(\chi_i)$. All of them are connected by an edge $e_{ij} = (\chi_i, \chi_j)$ to the seed node together with their correlation coefficient $\rho(\boldsymbol{x}_0, \boldsymbol{x}_i)$ as similarity measure s_{ij}. For all of

[1] The enlargement to more examples is straightforward. The example could be given by the user or by an appropriate detector.

these found image patches we repeat this procedure recursively. The depth 'rec' of recursion is a parameter, which we choose per default as rec = 3 but can be set by the user. The procedure is summarised in Alg. 1.

Algorithm 1 Recursive buildup of similarity graph

```
Require: $\chi_0$, rec, $T_1$, $I$
Ensure: $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{S})$
 1: $R = [\rho(\boldsymbol{x}_0, \boldsymbol{x}(i,j))] \ \forall i, j$
 2: $\mathcal{X}$ = non-maximum-suppression($R$, $T_1$)
 3: add $\chi_0$ and $\mathcal{X}$ to $\mathcal{V}$ and establish $\mathcal{E}$ and $\mathcal{S} = \{\rho(\chi_0, \mathcal{X}_i)\}$
 4: depth of recursion = 1
 5: while depth of recursion < rec do
 6:     for k = 1,...,length($\mathcal{X}$) do
 7:         $R = [\rho(\boldsymbol{x}_k, \boldsymbol{x}(i,j))] \ \forall i, j$
 8:         $\mathcal{X}$ = non-maximum-suppression($R$, $T_1$)
 9:         add $\mathcal{X}$ to $\mathcal{V}$ and update $\mathcal{E}$ and $\mathcal{S} = \{\rho(\chi_0, \mathcal{X}_i)\}$
10:     end for
11: end while
```

Unsupervised clustering using a similarity graph Finding clusters is based on a reduced similarity graph. Buhmann and Hofmann (1994) study the pairwise clustering problem in the framework of maximum entropy estimation. They assume only pairwise dissimilarity values are available and achieve grouping the data into clusters by minimising the sum of dissimilarities between data of the same cluster. In (Hofmann and Buhmann, 1997) they upgrade their method by a deterministic annealing approach and use pairwise data clustering to segment textured images. We in a first instance perform the clustering by simple thresholding, which is certainly suboptimal, but may be replaced by a more rigorous procedure. Therefore, we cut all edges with weights $\rho < T_2$ and derive connected components of the graph. So the vertices $v_i \in \mathcal{V}$ of the reduced similarity graph $\mathcal{G}\left(\mathcal{V}, \mathcal{E}, \mathcal{S}\right)$ represent instances χ_i of class $\mathsf{c} \in \mathsf{C}$. Two vertices v_i and v_j are connected in case the similarity $s(\chi_i, \chi_j) = \rho(\boldsymbol{x}_i, \boldsymbol{x}_j) > T_2$ is larger than a threshold T_2, thus the existence of e_{ij} indicates a high similarity. As this threshold T_2 is chosen to be larger than T_1 not all vertices in G are connected. We assume instances in one connected component to belong to the same subclass.

Per default we choose the correlation coefficients $T_1 = 0.6$ and $T_2 = 0.8$ as thresholds. We are currently investigating the adequate choice of these parameters.

Supervised labelling of found clusters The identification of the subclasses is based on the user's judgement. For each cluster the user has to specify whether it represents a subclass of the class envisaged, another class not yet envisaged, or the rejection class. The user thus establishes a simple class hierarchy with multiple classes C_k, including the rejection class, with the subclasses C_{ki}. E. g. we may take the class window as the envisaged class, manually identify a window by its bounding box and - as an example - may have found four clusters, which turn out to be of different type, e. g. windows with and without crossbar and with a round top, and another cluster of balconies, see Fig. 2(a).

Representation of prototypes We assume that our clustering and labelling process generates well closed clusters for each class. Thus we may assume the p. d. f. to have one mode, or at maximum a few. We might use the mean feature vector as prototype and characterize it by its covariance matrix. The chosen algorithms for finding prototypes need to be seen within the complete scheme of incremental learning, which poses restrictions on the representation which need to be updatable incrementally.

We determine a prototype by finding a representative instance as a function of the found instances. In our first realization we use the mean images as prototypes for each class. So we define the prototype $\boldsymbol{p}_c$ of each class $c \in \mathcal{C}$ as

$$\boldsymbol{p}_c = \overline{{}^c\boldsymbol{x}} = \frac{1}{N_c}\sum_{n=1}^{N_c} {}^c\boldsymbol{x}_n \qquad (1)$$

where ${}^c\boldsymbol{x}_i$ are the feature vector of instances ${}^c\chi_i$ of class C and N_c is their number.

To achieve invariance on different object sizes we scale our prototypes to a given fixed size.

Experiments with getting prototypes Fig. 2(a) shows a facade with four floors and five window columns. We observe two classes of facade elements: windows and balconies. The windows can be partitioned into four subclasses. We add the cornices to the windows as a characteristic feature. We chose a bounding box around the second window on the third floor to define a single example. To reduce the computational costs we resized the image, so that the templates got size 50×70 pixels. We took our default parameter set with recursion depth of three and $T_1 = 0.6$ and $T_2 = 0.8$ as thresholds for correlation coefficients.

Fig. 2(b) shows the mean images of found clusters together with labels given by the user manually, see figure caption for explanation the numbers. Obviously, the joint p. d. f. of features has such a form that the chaining effect allows a drift of the found image patches to different clusters. The third found cluster (third image at first line) shows that the cornices were identified as a cluster of multiple occurring elements. Indeed, this element is the most frequent element in image. But as we are not interested in such objects we chose this cluster to be background. As a further effect of the chaining effect we reached the desired drift to not only different types of the same super class but also to a new super class. Hence, the process identified two new window subclasses. And as the balconies are very similar to the windows, the process also identified them as related class.

Fig. 2(c) shows found image patches marked with their label. Nearly all interesting instances were found excepting three windows on the upper floor. The rectangular windows without cornice could not be distinguished from those with cornices. But all others were correctly found and matched to their class. Finally, Fig. 2(d) shows the corresponding object hierarchy with classes represented by their prototypes.

4.2 Classification

Models for classification can be generative or discriminative. Generative models capture the essence of the class explicitly and allow sampling of instances, e. g. feature vectors from their compactly represented distribution. Incremental learning of generative models is incrementally changing the class description. Discriminative models explicitly represent differences between two or more classes. They make the decision process explicit. Incremental learning is changing the decision rules, e. g. the decision function. Incremental learning is much easier in generative models. The efficiency of discriminative models in general is much higher. This is the reason why both representations should be integrated, cf. (Skočaj et al., 2006). In a first step towards incremental learning we follow the approach of Fidler (2006) and use augmented PCA as generative model for our classes and LDA for representing the decision functions.

This procedure is called LDA on augmented PCA (LDAaPCA) and as result we obtain feature vectors $\boldsymbol{y}$ for every image patch in the final classification subspace which is of dimension $C - 1$.

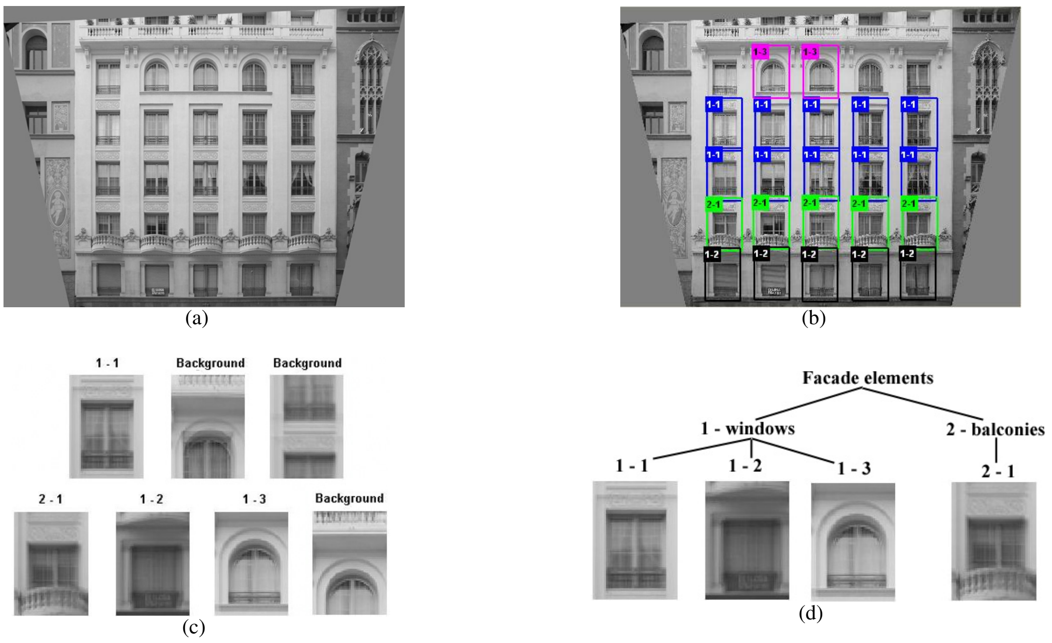


Figure 2: (a): Rectified facade image with two super classes, windows and balconies and four different subclasses of class window. (b): Found image patches marked with their label. (c): Mean images of found clusters and their labels given by the user manually. The first number codes the class, here 1 for window and 2 for balcony and the second number codes the type and subclass, respectively, here three different window types. (d): Object hierarchy, with classes represented by their prototypes.

The sample is matched to that class c for which the a posteriori probability according a bayes classificator is maximised.

$$p(\boldsymbol{y}|c) = \text{argmax}_c\, p\left(\boldsymbol{y}|\mu_c, \Sigma_c\right) p\left(c\right) \tag{2}$$

We assume an uni-modal gaussian $p\left(\boldsymbol{y}|\mu_c, \Sigma_c\right)$ with mean μ_c and covariance Σ_c for every class c. The a priori probability $p\left(c\right)$ for every class is simply the fraction of the number N_c of samples of the class c to the whole number of samples N, $p\left(c\right) = N_c/N$. This way we can introduce a reject option for which the classification is uncertain.

$$p_\alpha\left(\boldsymbol{y}\right) = 1 - \frac{p\left(\boldsymbol{y}|c_{p_{max}}\right)}{\sum_{c=1}^{C} p\left(\boldsymbol{y}|c\right)} \tag{3}$$

The reject option holds if $p_\alpha > \varepsilon$ for which we choose a significance threshold of $\varepsilon = 0.01$. In that case the according image patch is presented to the user and he has to decide whether to accept the sample or to reject it.

4.3 Detecting candidates for new instances in new images

To detect new instances in new images we use almost the same procedure as described in Sec. 4.1. As we work on metric images we assume the image scale given. Thus, we rescale the image according to the given prototype scale. We then use the prototypes for detecting at least one new instance in the new image and start the recursive search procedure described in Sec. 4.1 to detect probable new instances. In contrast to Sec. 4.1 we now pass on the clustering and classify all found instances according to the learned classification models.

4.4 Incremental update of prototypes and classifiers

First we need to update our prototypes $\boldsymbol{p}_c$ for detecting new instances in new images. This is done by simply updating the already known mean images $\boldsymbol{p}_c^\nu$ of class c of last step ν by adding the mean $\boldsymbol{p}_c$ of new images.

$$\boldsymbol{p}_c^{\nu+1} = \frac{N_c^\nu \boldsymbol{p}_c^\nu + N_c \boldsymbol{p}_c}{N_c^\nu + N_c} \tag{4}$$

whereas N_* are the associated numbers of samples.

As we have only few examples per class, in our first implementation we evaluate LDAaPCA on all data gathered during the recognition phase. Thus, after receiving a new example the subspace is updated as well as the coefficients of all images seen before. To increase the performance we will adapt it to an incremental LDA, cf. (Uray et al., 2007) which is a combination of an incremental PCA on an augmented PCA subspace and the LDA on this updated aPCA space. This way we will be able to handle a long sequence of images and to continuously update our class models.

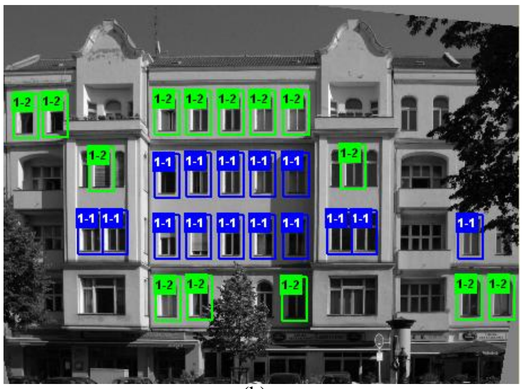
5 EXPERIMENTS

Now we want to describe an experiment where we detected and classified windows over a sequence of images given one example and built up the class hierarchy.

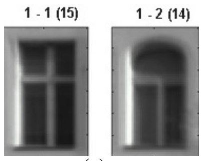
Fig. 3(a) shows a rectified facade image where we gave the system one example of a window, marked red. Given this example we started the recursive search and clustering procedure and found new window instances, shown in Fig. 3(b). The second class, marked green, was established after asking the user for the meaning of this cluster. Thus the initial class hierarchy consists of one root node, class 1 (window) with two leafs 1-1 and 1-2. Fig. 3(c) shows the associated prototypes for these two classes. And Fig. 3(d) shows the samples projected onto the subspace together with the class decision boundaries which are indented from the class boundaries by the rejection area. A sample projected onto the region between the decision boundaries can not be reliably matched to one class, hence the user is asked for the meaning of this sample.

Next images Fig. 4 - 6 shows the process of detecting new instances and updating the learned models. First we took the learned

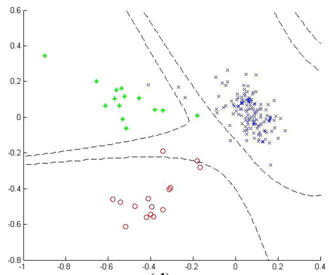
(a)



(b)

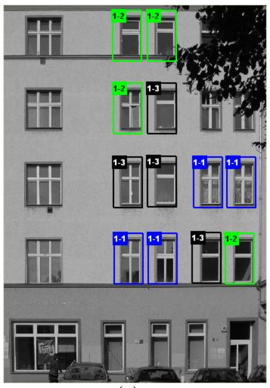


(c)

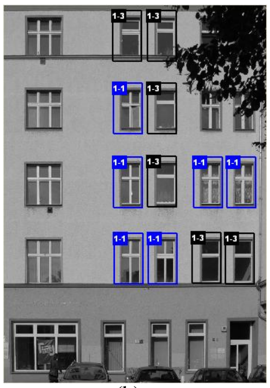


(d)

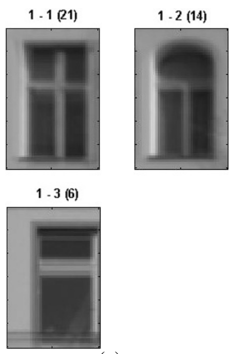
Figure 3: Initialisation: (a) Initial example. (b) Result of detecting new instances. (c) Prototypes. (14): Number of associated samples. (d) Samples projected into the LDA subspace. Red: class 1-1, green: class 1-2, blue: background. Dashed lines: decision boundaries according the class boundaries and the rejection area.



(a)

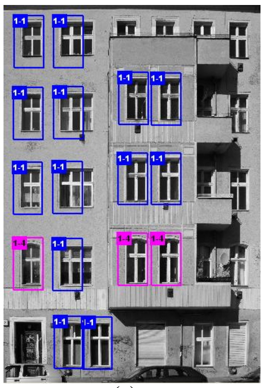


(b)

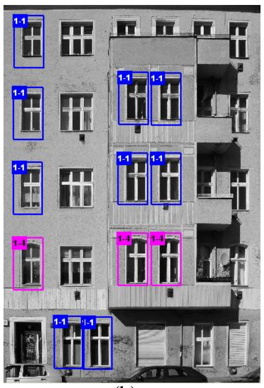


(c)

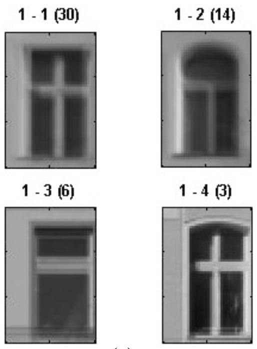
Figure 4: 2. Step: Automatically detect new instances. (a) Result before editing. (b) Result after editing. (c) Prototypes.



(a)



(b)



(c)

Figure 5: 3. Step: (a) Result before editing. (b) Result after editing. (c) Prototypes.

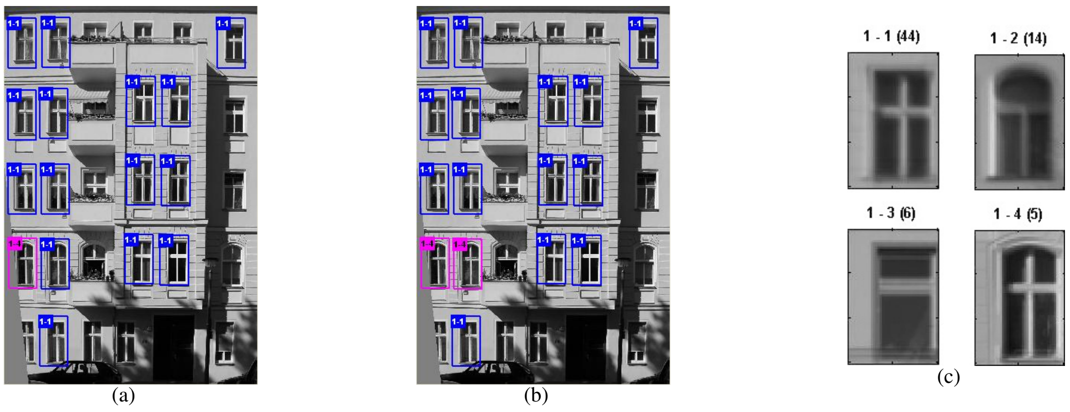


Figure 6: 4. Step: (a) Result before editing. (b) Result after editing. (c) Prototypes.

prototypes from the initialisation and used them to find at least one new instance in the facade image shown in Fig. 4. Than we recursively searched for all probable new instances and classified them according to the learned models. Instances that could not be clearly matched to one class were presented to the user. This way a new class 1-3 was established, marked black. The result is shown in Fig. 4(a). Of course, the classifiers were not very robust at this stage as they were learned from only a few examples. Hence there were some misclassifications that were manually corrected by the user. The result after editing is shown in Fig. 4(c). The image patches shown were used to update the class hierarchy, that is, to update already known prototypes and the initialisation of new classes as well as to update the subspace representations. As the dimension of the feature subspace increases with every new class, we pass on the presentation of features in subspace like Fig. 3(d). Note, that we started with only one example of a certain size and height-width-ratio, respectively. Thus we did not recognise the bigger windows. This would be fixed when defining a new example of a different size.

Fig. 5 and 6 shows the results of step 3 and 4 the same way. Step 3 has established a new subclass 1-4. A sample of this class was detected and correctly classified within the next image. As instances of class 1-1 occur in every image - there were 30 instances found within the first three images - the classifier became more robust. Hence the classification results for the image of Fig. 6 are quite better than for the first ones.

6 CONCLUSIONS AND FUTURE WORK

We gave a concept for an incremental learning scheme. Given one example of an object within a rectified image and given the prior knowledge that the object appears several times in the image, we learn the variation in appearance of the class of the given object to detect further instances in other images. We have shown a recursive procedure to find similar objects within an image. By an unsupervised clustering we are able to make a hypothesis about different object classes within an image. Finally, by minor help of the user we identify new object classes or new subclasses among the found clusters. That way, we build up an object hierarchy of classes and subclasses with minimal user interaction that is updated with every new image.

The results of the recursive search and the clustering procedure up to now depend too much on the choice of the thresholds T_1 and T_2. We will either use an optimisation procedure to find best thresholds T_1 and T_2, e. g. using a hierarchical clustering procedures, such as dendrogramms, to find an optimal threshold for T_2. Or, as an alternative, we might use a more sophisticated clustering procedure which contains an optimization function, thus avoiding the need for setting thresholds.

To increase the classification performance we will adapt the current subspace methods to an incremental LDA, cf. Uray et al. (2007). However, we get a feature vector that can be expanded by further features, e.g. depth information, obtained from surface reconstruction given prior knowledge of symmetry, which we can assume for objects like windows and balconies, cf. (Hong et al., 2004), (Yang et al., 2005) and then can be used for increasing classification performance.

ACKNOWLEDGEMENTS

This work is founded by the EU-Project 027113 eTRIMS, E - Training for Interpreting Images of Man-Made Scenes.

References

Belhumeur, P. N., Hespanha, J. and Kriegman, D. J., 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. PAMI 19(7), pp. 711–720.

Buhmann, J. M. and Hofmann, T., 1994. A maximum entropy approach to pairwise data clustering. In: ICPR, pp. 207–212.

Fei-Fei, L., Fergus, R. and Perona, P., 2004. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In: CVPR Workshop on Generative-Model Based Vision.

Fidler, S., 2006. Combining Reconstructive and Discriminative Subspace Methods for Robust Classification and Regression by Subsampling. PAMI 28(3), pp. 337–350.

Hofmann, T. and Buhmann, J. M., 1997. Pairwise data clustering by deterministic annealing. In: PAMI, Vol. 19, pp. 1–14.

Hong, W., Yang, A. Y., Huang, K. and Ma, Y., 2004. On symmetry and multiple-view geometry: Structure, pose, and calibration from a single image. IJCV 60(3), pp. 241–265.

Li, L.-J., Wang, G. and Fei-Fei, L., 2007. OPTIMOL: automatic Object Picture collecTion via Incremental MOdel Learning. In: CVPR.

Sivic, J. and Zisserman, A., 2006. Video Google: Efficient visual search of videos. In: Toward Category-Level Object Recognition, LNCS, Vol. 4170, Springer, pp. 127–144.

Skočaj, D., Uray, M., Leonardis, A. and Bischof, H., 2006. Why to combine reconstructive and discriminative information for incremental subspace learning. In: CVWW 2006, Telč, Czech Republic.

Uray, M., Skočaj, D., Roth, P. M., Bischof, H. and Leonardis, A., 2007. Incremental LDA Learning by Combining Reconstructive and Discriminative Approaches. In: BMVC.

Van Gool, L., Zeng, G., Van den Borre, F. and Müller, P., 2007. Towards mass-produced building models. In: PIA, Vol. 36, Munich, Germany, pp. 209–220.

Yang, A. Y., Huang, K., Rao, S., Hong, W. and Ma, Y., 2005. Symmetry-based 3-d reconstruction from perspective images. In: CVIU, Vol. 99number 2, pp. 210–240.