An efficient multi-resolution SVM network approach for object detection in aerial images

Jérôme Pasquet, Marc Chaumont, Gérard Subsol, Mustapha Derras

To cite this version:

Jérôme Pasquet, Marc Chaumont, Gérard Subsol, Mustapha Derras. An efficient multi-resolution SVM network approach for object detection in aerial images. MLSP: Machine Learning for Signal Processing, Sep 2015, Boston, United States. 10.1109/MLSP.2015.7324329 . lirmm-01234225

HAL Id: lirmm-01234225
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01234225v1

Submitted on 26 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

2015 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2015, BOSTON, USA

AN EFFICIENT MULTI-RESOLUTION SVM NETWORK APPROACH FOR OBJECT DETECTION IN AERIAL IMAGES

J. Pasquet[⋆†] *M. Chaumont*[*†] *G. Subsol* [†] *M. Derras*[⋆]

[⋆] Berger Levrault, Labège, France
[†] LIRMM, Université de Montpellier / CNRS, France
[*] Université de Nîmes, France

ABSTRACT

In this paper, we deal with the problem of object detection in aerial images. A lot of efficient approaches uses a cascade of classifiers which process vectors of descriptive features such as HOG. In order to take into account the variability in object dimension, features at different resolutions are often concatenated in a large descriptor vector. This prevents from taking into account explicitly the different resolutions but results in losing some valuable information.

To overcome this problem, we propose to use a new method based on a SVM network. Each resolution is processed, regardless to the others, at the input layer level, by a dedicated SVM. The main drawback of using such a network is that the computational complexity for the classification phase drastically increases. We propose then to foster an incomplete exploration of the network by defining an *activation path*. This activation path determines an order to activate the network neurons, one after the other, and introduces a rejection rule which allows the process to end before crossing the whole network.

Experimental results are obtained and assessed in an industrial application of urban object detection. We can observe an average gain of 17% in precision while the computational cost is divided by more than 5, with respect to a standard method.

***Index Terms*—** Network, Multi-Resolution, Object detection, SVM, Aerial Imagery

1. INTRODUCTION

In computer vision, object detection in natural images is a major challenge [1]. The goal is to be able to detect the presence or not of an object and to delineate its boundaries or, at least, a bounding box around it. Many practical applications can be found as face recognition in photography [2, 3], pedestrian detection [4, 5] or car detection in aerial imagery [6]. A seminal paper by Viola and Jones was published [2] in 2001, where the classifier, based on Haar-like features, detects efficiently faces. The authors propose then to reduce the complexity of the learning step by using Adaboost [7, 8] where they select the best subsets of features to create a so-called cascade architecture which combines different classifiers. This cascade has the form of a degenerated tree [9] where each classifier is activated in terms of its preceding classifier. Such an approach reduces the computational time by 15 [2], allowing the application to be real-time.

Since 2001, there is two main trends in object detections. The first one introduces efficient feature [10] as Histograms of Oriented Gradient (HOG) [11] which are now widely used. They characterize the distribution of gradient intensity and direction in the object, which is closely related to contours defining the object shape. More recently, Zhu and al. [4] proposed to extract and concatenate HOG from different resolutions in a multi-resolution HOG vector. But in this case, we may lose some valuable information about resolution when we concatenate all resolutions into a single feature vector. Moreover, the Adaboost step does not take into account the resolution as it searches only direct strong subset. An important issue is then to propose a new way to integrate explicitly the multi-resolution features in the learning process.

The second trend uses an efficient architecture to combine classifiers. In particular, the so-called Deep Learning methods are based on an actual network of classifiers [3, 12, 13]. In that case, we do not have a cascade structure: we do not activate the classifiers in the ascending order of complexity and objects are no more generally detected by low-complexity classifiers, leaving high-complexity classifiers analyzing a minority of cases. This requires then to find a new method of activation of the network to get efficient algorithms.

The rest of the paper is organized as follows : first, we propose in section 2 to integrate a multi-resolution descriptor based on HOG features in a SVM network in order to really combine the different levels of resolution. Then, in section 3, we explain how to reduce the complexity of the obtained SVM network with an efficient activation scheme. In section 4, we present some experiments to evaluate the performance of our multi-resolution architecture for an object detection application in aerial imagery. Finally we present some tracks for future research in section 5.

978-1-4673-7454-5/15/$31.00 ©2015 IEEE

2. AN SVN NETWORK BASED ON MULTI-RESOLUTION DESCRIPTORS

2.1. Multi-Resolution Descriptor Definition

In order to get a robust multi-scale descriptor, we normalize the bounding box of each object of the training database to an image of a constant size as in [2]. Then, we extract the HOG [11] feature in a sliding window on the normalized image of the object. For an object, we obtain a vector of HOG features and to get a multi-resolution descriptor, we extract the HOG vector with different sizes of sliding window as in [4]. We define the set of resolutions of extraction $\mathbf{R} = \{0, .., R\}$ with R the number of resolutions. We note $f_j^{(r)}$ the feature vector HOG with the resolution $r \in \mathbf{R}$ and j the sliding window position. At the end, for one normalized image we obtain R HOG vectors, noted $f^{(r)}$, where $f^{(r)}$ is the concatenation of all $f_j^{(r)}$ from all the sliding window positions with $r \in \mathbf{R}$. Figure 1 resumes the extraction process.

Usually, the R HOG vectors are concatenated in a single huge vector, noted f [4] which is the input vector for the learning process. During the testing step, extraction and classification on the full vector f, for each testing window, takes a lots of time. To reduce the complexity we can use the cascade of windows rejector approach. Viola and Jones [2] propose to identify a *strong* features subset with an AdaBoost algorithm [8]. A drawbacks of Adaboost is that the strong subset may merge several features from different resolutions, losing information about a specific resolution. In order to avoid this problem, we propose to train several classifiers on each resolution and to combine all these classifiers using an SVM network. We thus build an SVM network were each neuron is an SVM classifier.

2.2. Building a Network of Linear SVM in order to Integrate Multi-Resolution

The presented approach does not use only the single vector f but distinguishes the information coming from the different resolutions, by a network using. Usually, a network is used to combine different features into a same framework without normalization problem. For example, we can easily concatenate HOG with the Haar-like features or the Speeded Up Robust Features (SURF) [14]. For this reason, neural network is often used to treat detection problems [3].

We propose to treat more specifically each resolution by feeding an SVM neuron with a resolution $f^{(r)}$, $r \in \mathbf{R}$. For each resolution in $\mathbf{R}$, an input neuron takes the vector $f^{(r)}$ as shown in Figure 1. A series of hidden layers are then randomly connected to previous layer. The output network neuron is finally full connected to all neurons from the last hidden layer.

Support vector machine (SVM) method has already been successfully used to solve detection problems. Each linear SVM [15] will then be considered as a neuron [12, 16, 17].

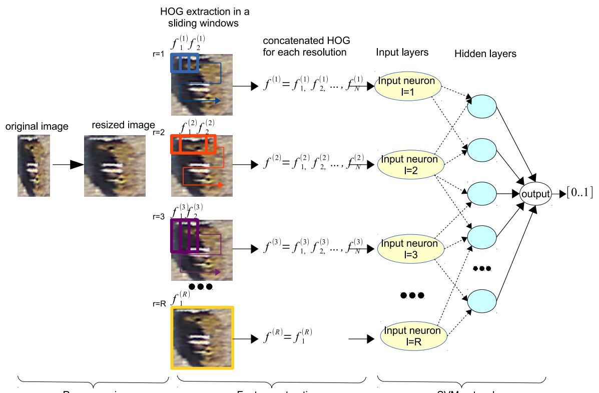

Fig. 1. Schema of network inputs to our detection algorithm.

During the learning, we train sequentially each SVM-neuron starting from the first layer until the last layer. Each linear SVM computes a vector known as the weight vector, noted $w^{(l)}$, with $l \in \{0..L\}$ the number of SVM-neuron and L the size of the network. Once learned, each linear SVM takes a vector $f^{(l)}$ as an input and return the dot product with $w^{(l)}$; we denote this local classification $s^{(l)} \in \mathbb{R}$:

$$s^{(l)} =< f^{(l)}, w^{(l)} > \tag{1}$$

Beside, an additional treatment is applied to the classification score $s^{(l)}$ before returning this value to the son of the SVM-neuron l. This treatment consists of applying an activation function to the dot product $s^{(l)}$ which its purpose is to break the linearity. We define $p^{(l)} \in [0..1]$ the activation function, such as a sigmoid function, associated with the SVM-neuron l :

$$p^{(l)} = (1 + e^{-\alpha^{(l)} s^{(l)} + \lambda^{(l)}})^{-1} \tag{2}$$

Where $p^{(l)}$ the output probability of the SVM-neuron number l, $\alpha^{(l)}$ and $\lambda^{(l)}$ are two parameters computed with a validation database according to [18]. The output SVM-neuron has an activation function $p^{(L-1)}$ that gives the final probability for the network to have found the interested object.

Thus, in our proposed approach each input SVM-neuron takes only one resolution. Nevertheless, we can also create efficient input SVM-neurons with other combinations of features. For this purpose, we search for the best subsets of features. This can be done by two ways: we can use a selection method like Adaboost or we can just randomly choose a subset of the full feature vector f as in [19]. The problem with a selection method is that we can find only the strong subsets of features for the input SVM-neurons without taking account the hidden layers. So we prefer the second solution which could give better results using the network. We define a *random neuron* as an SVM-neuron in the input layer with a random subset of features. Each *random neuron* is randomly connect to the hidden layer as the input SVM-neurons.

3. AN EFFICIENT ACTIVATION SCHEME TO SPEED-UP PROCESSING

3.1. Complexity of a Network

During the evaluation, the entire SVM network is activated in order to give a probability for a query window. The computational complexity is then proportional to the sum of the dimension of each vector in input of all the neurons [20]. Even if the computational complexity is linear, we need to activate few billions of windows for our entire SVM network [21].

The computational cost is so expensive and the classifier cannot be used for any application without the cascade reject system [2]. Another solution would consist to reduce the number of testing windows with a preselection [22] but there is a risk of forgetting some good windows.

To tackle the issue, we propose to adapt the cascade reject system into our SVM network. Each neuron is consequently activated in a predefined order and thus the full network does not necessary require an entire activation.

3.2. Speeding Up the Network

In order to classify a given input feature vector, we propose, to traverse the network in a cascade manner and stop the exploration as soon a decision can be done. Once a traverse path has been built, given an input vector, each neuron is activated sequentially, i.e. each neuron (l) outputs a value $p^{(l)}$ and we select each neuron in the order defined by the traverse path. If the output $p^{(l)}$ given by the neuron, after activation, is lower than a precalculated threshold $\theta^{(l)}$, the window is rejected. Nevertheless, if the output given is higher than a precalculated threshold $\theta^{(l)}$ we activate the next neuron, as shown in Figure 2.

The arising question is how to build the traverse path (also named *activation path*)? The process is iterative and consists, for each iteration, to find the *best next neuron* to activate. The *best next neuron* belongs to the set of neurons which are connected to the neurons belonging to the path, i.e. previously activated. In order to find this *best next neuron*, we define a cost function to associate a cost to each neuron and then select the neuron with the lower cost to be the *best next neuron* and is defined for neuron l as:

$$c^{(l)} = dim(f^{(l)}) + \sum_{p \in Fathers(l)} c^{(p)} \tag{3}$$

With dim the function that gives the dimension of a vector and $Parent(l)$ the set of neurons which are the fathers of the neuron (l). In the figure 2, we have $dim(f_1) < dim(f_2) < dim(f_3)$ and thus $c^{(A)} < c^{(B)} < c^{(C)}$. The first two neurons add in the activation path are the neuron A and the neuron B. In this example, we also assume $dim(f_1) + dim(f_2) + 2 < dim(f_3)$ which implies that $c^{(D)} < c^{(C)}$ and therefore the neuron D has the third position in the activation path. The following activated neurons are C, E and F.

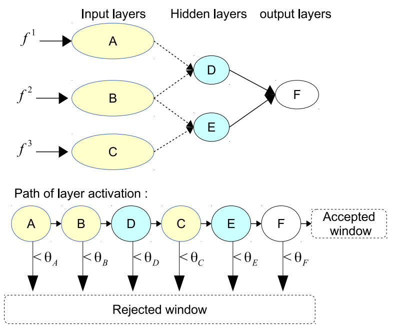

Fig. 2. A small network and its activation scheme.

Now, given an *activation path*, we have to set the threshold value $\theta^{(l)}$ that is used for each output $p^{(l)}$ in order to decide if each neuron rejects the input vector or if it activates the next neuron in the *activation path*, as in equation 4 :

$$\forall l \in \{1..L\}, if \begin{cases} p^{(l)} \leq \theta^{(l)} & \rightarrow \text{stop the traverse} \\ p^{(l)} > \theta^{(l)} & \rightarrow \text{activate the next neuron} \end{cases} \tag{4}$$

The threshold $\theta^{(l)}$ should be defined to minimize the number of false negatives and thus do not reject any object. Using a validation database for each neuron we fix the threshold $\theta^{(l)}$ to maximize the recall, under the constraint to have a precision greater or equals to the minimal value $p^{(l)}_{min}$, see equation 5 :

$$\theta^{(l)} = \arg\max_{\theta \in [0..1]} \mathit{Recall}^{(l)}(\theta) \text{ such that } \mathit{Precison}^{(l)}(\theta) \geq p^{(l)}_{min} \tag{5}$$

with $\mathit{Recall}^{(l)}(\theta)$ and $\mathit{Precison}^{(l)}(\theta)$ the functions giving the recall and precision of neuron l for a θ threshold fixed.

Note that there is two strategies to determine the value $p^{(l)}_{min}$. Firstly, we may consider $p^{(l)}_{min}$ as a constant value. Secondly, we may consider that $p^{(l)}_{min}$ value increases with the computational cost of the neuron. The second approach is close to the system of cascade reject. For example, in the Viola and Jones approach [2], the more we use models for detection, the more we increase the precision but the recall decreases.

When we search for the best threshold $\theta^{(l)}$ the neuron l has a precision higher than the minimal precision $p^{(l)}_{min}$. Under this condition, the recall of the neuron l may be low and so it will reject a lot of feature vectors. We define neurons which have a recall lower than 90% as bad neurons. These bad neurons are not added to the activation path and they cannot reject any windows. However, they can be activated to give their output $s^{(l)}$ to their sons. In fact, the activation path does not contain the L neurons from the network but only a subset with the best of them.

To resume, after the learning of the network, activation path is built. This activation path is the ordered list of neuron numbers defining the order of the network. Additionally, a threshold $\theta^{(l)}$ is also computed for each neuron. The threshold allows to define a stop criterion in order to stop the network exploration during the testing phase. $p^{(l)}_{min}$ values are also set to determine the flexibility of each neuron.

4. EXPERIMENTAL RESULTS AND EVALUATION OF THE METHOD

4.1. Description of Image Data

In this application, we are interested to the detection of urban objects in high-definition aerial imagery. More specifically, we focus on detecting tombs in cemetery images for geo-localization and digital heritage purposes [23]. Tomb detection is a very challenging problem as tombs vary substantially in appearance, color, size and disposition on aerial images. Moreover, vegetation, shadows created by the numerous buildings, walking people or utility vehicles may create many distortions and occlusions in the images. At last, this is multiple object detection as cemeteries contain many tens or thousands tombs.

To assess our method we use a database of 24 high-resolution (2.5 cm/pixel) aerial images with size about 5000 x 5000 pixels of French cemeteries given by the Berger Levrault company [1].

For the training database we use 19 images, which contain about 4,500 tombs. For the validation database we use 2 images, about 700 tombs. To evaluate our algorithm we use 3 cemetery images (about 750 tombs) where we have manually delineated the rectangular bounding boxes of tombs which will be considered as ground truth.

During the evaluation step, detection results are given as a list of rectangular bounding boxes. To be considered as a correct detection, the area of overlap, noted A (see eq 6), between the detected boxes B_d and the ground truth B_g must exceed 58%.

$$A = Area(B_d \cap B_g).Area(B_d \cup B_g)^{-1} \qquad (6)$$

4.2. Finding Optimal Parameters for Network Topology

A lot of parameters can affect the final probability given by classifier. It is a hard problem to optimize the network because we need to find the best number of random and hidden neurons [24]. In this paper, we propose to use simulations which proceed in two steps. Firstly, we determine the number of hidden neurons. Secondly, we search for the best number of *random neurons*. We use the High Performance Computing resources of HPC-LR[2] in order to find the best number of hidden and random neurons in a reasonable computational time. In Figure 3, we observe the network becomes more robust with the increasing number of hidden neurons. However, after using more than 300 hidden neurons the efficiently of the network decreases by 3.2%. This performance drop is caused by the over fitting of the network on the training database. In our case we find the best combination of parameters is about 300 hidden neurons and 200 random neurons.

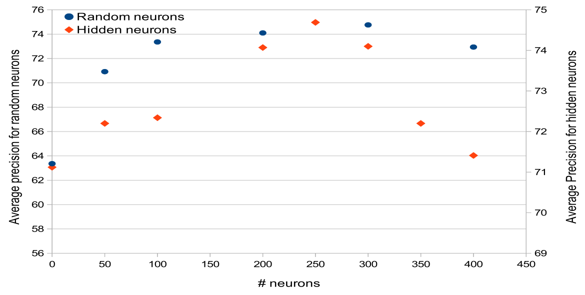

Fig. 3. Average of the precision compute with the validation database for a recall between 45% and 80% as function of the number of hidden and random neurons.

4.3. Evaluation of the Multi-Resolution Architecture

Figure 4 illustrates the performance of the method without network i.e. only one SVM which takes the full feature vector f and the method using SVM network. The two methods have a maximum recall of 85% because in the window fusion phase we delete windows with lower probabilities. So even if we set a low detection threshold we do not detect all graves.

The performance of the two methods differs in the way to converge. Indeed, the method with network is more efficient for a recall range from 45% to 80%. For example, we notice for a recall of 75% the precision with our method is increase by 17%. Using a network with few resolutions as inputs outperforms the approach using only one SVM model. This can be explained by the fact that SVM network using a sigmoid activation function create a non-linear output compared to a single linear SVM.

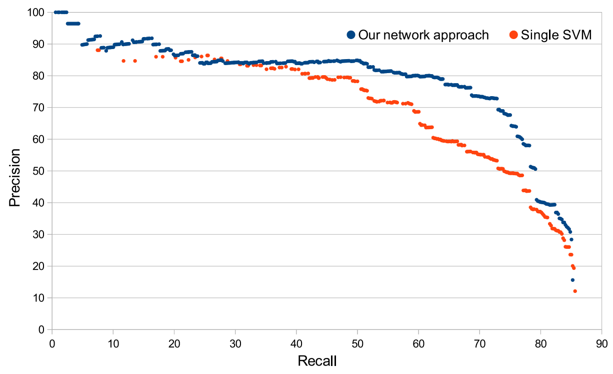

Fig. 4. The orange curve shows the performance of the method with a single SVM. The blue curve gives the efficiency of our SVM network approach.

[1]Berger-Levrault is a French public regulation expert that addresses healthcare and local public administrations, www.berger-levrault.com

[2]HPC@LR: High Performance Computing from Languedoc-Roussillon, https://www.hpc-lr.univ-montp2.fr

4.4. Evaluation of the Activation Path

Approaches	time (HH:MM)	average precision
Single SVM	15:54	60.6%
Our SVM network	24:35	75.0%
Activation path p_{min} increases	04:58	75.3%
Activation path p_{min} constant	05:48	74.5%

Table 1. Performance and computational cost during the evaluation step for the different methods on three images.

As stated, the main problem of network approach is the computational cost. Indeed, as we can see on Table 1 that the time requirements for the evaluation step is huge with an average of 24 hours 35 min by image. This prevents any practical use.

Consequently we need to use the speeding up network. In the table 1, we give performance of system with path in two scenario: - the p_{min} value increases from 1% to 25% ; - p_{min} is a constant value set to 5%. We can see there is no major difference in efficiently between the network which is always full activate and the network which using an activation scheme. Indeed, the average precision of the full network is around 75.0% against 74.5% and 75.3% for networks using an activation scheme. One day of computational performance is required to activate the network without any optimization. However, using an activation scheme reduces the time taken to 5.5 hours, thus a reduction by 4. In addition, the speeding up network used 2.5 times less than only one SVM.

Using the table 1 we examine the best strategy to build the activation scheme. This table shows that the scenario with a constant minimal precision p_{min} takes more time than the scenario with an increasing p_{min}. Indeed, in the increasing scenario the minimal precision p_{min} increases until 25% and so the network rejects more windows. This explains the time computational required is reduced by 3,000 seconds. We can thus predict if the constant minimal precision increases, the time required decreases. As we can see from Figure 5, the computational times is converging to a computational time close to 15,000 for a p_{min} value greater than 8%. However, if the minimal precision p_{min} is too high the time required converges because the number of bad neurons increase with the value p_{min}. Finally, for a p_{min} value higher than 8%, the constant scenario is the more cost effective than the increasing scenario and we reduce the computational time by 5.5. The constant scenario is more efficiently but it needs a validation database to fix the p_{min} to 8%.

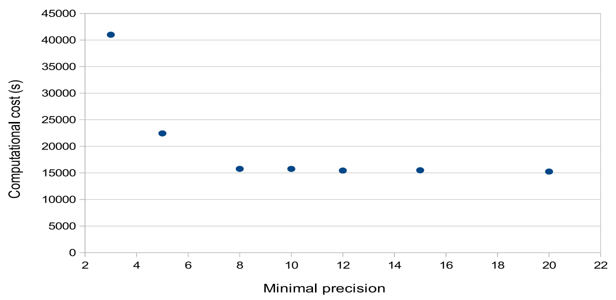

Fig. 5. Computational cost for the speed up network as function of the constant p_{min} value with an average precision around 75%.

5. CONCLUSION AND PERSPECTIVES

In this paper, we presented a SVM network which takes into account full information from each resolution. For each resolution, we extract an HOG feature for all positions and during the learning step, we process each resolution regardless to the others input nodes of our network. Then, the hidden layers allow to combine the outputs of each input layer to get the best non-linear network. We show that, this kind of approach increases the precision by 17%.

Nevertheless, such a network is very complex since there are many nodes and the computational cost increases by 50% compared to a single SVM. To reduce the complexity we propose to activate each layer in a predefined order which defines an *activation scheme*. This method allows a partial activation of the entire network and reduces the computational time by a 5.5 factor in average.

Future work will deal with adding some features. For example, we could introduce some shape descriptors, such as SURF [14], to detect a cross pattern on the grave. Indeed, when we divide the vector into few input layers we allow to use different types of feature without any normalization problem. Another extension of our work could be to extend the activation path to an huge network of several thousand neurons as a convolutional neural networks [25].

6. REFERENCES

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, June 2010.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, vol. 1, pp. I–511–I–518 vol.1.

[3] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade, "Human face detection in visual scenes," in *Advances in Neural Information Processing Systems 8*, D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, Eds., pp. 875–881. MIT Press, 1996.

[4] Qiang Zhu, Qiang Zhu, Shai Avidan, Shai Avidan, Mei chen Yeh, Mei chen Yeh, Kwang ting Cheng, and Kwang ting Cheng, "Fast human detection using a cascade of Histograms of Oriented Gradients," in *CVPR06*, 2006, pp. 1491–1498.

[5] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, June 2009, pp. 304–311.

[6] Helmut Grabner, Thuy Thi Nguyen, Barbara Gruber, and Horst Bischof, "On-line boosting-based car detection from aerial images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 63, no. 3, pp. 382 – 396, 2008.

[7] Yoav Freund and Robert E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[8] Yoav Freund and Robert E. Schapire, "A short introduction to boosting," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. 1999, pp. 1401–1406, Morgan Kaufmann.

[9] Yali Amit and Kenneth Wilder, "Joint induction of shape features and tree classifiers.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 11, pp. 1300–1305, 1997.

[10] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002, vol. 1, pp. I–900–I–903 vol.1.

[11] Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for human detection," in *CVPR*, 2005, pp. 886–893.

[12] Xingyu Zeng, Wanli Ouyang, and Xiaogang Wang, "Multi-stage contextual deep learning for pedestrian detection," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 121–128.

[13] Shiliang Sun, "Ensembles of feature subspaces for object detection," in *Advances in Neural Networks ISNN 2009*, Wen Yu, Haibo He, and Nian Zhang, Eds., vol. 5552 of *Lecture Notes in Computer Science*, pp. 996–1004. Springer Berlin Heidelberg, 2009.

[14] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, June 2008.

[15] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[16] O. Vinyals, Y. Jia, Li Deng, and Trevor Darrell, "Learning with recursive perceptual representations," *Neural Information Processing Systems (NIPS)*, vol. 15, December 2012.

[17] Yichuan Tang, "Deep learning using support vector machines," *Computing Research Repository (CoRR)*, vol. abs/1306.0239, 2013.

[18] Jing Gao and Pang ning Tan, "Converting output scores from outlier detection algorithms into probability estimates," in *In Proc. of the Sixth IEEE Int. Conf. on Data Mining (ICDM06)*, 2006, pp. 212–221.

[19] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun, "What is the best multi-stage architecture for object recognition?," in *Proc. International Conference on Computer Vision (ICCV'09)*. 2009, IEEE.

[20] M.A. Kon and L. Plaskota, "Information complexity of neural networks," in *Neural Networks 13*, 2000, vol. 1, pp. 365–376.

[21] David Aldavert, Arnau Ramisa, Ricardo Toledo, and Ramon López de Mántaras, "Fast and Robust Object Segmentation with the Integral Linear Classifier," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1046–1053.

[22] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[23] M. Chaumont, L. Tribouillard, G. Subsol, F. Courtade, J. Pasquet, and M. Derras, "Automatic localization of tombs in aerial imagery: Application to the digital archiving of cemetery heritage," in *Digital Heritage International Congress (DigitalHeritage), 2013*, Oct 2013, vol. 1, pp. 657–660.

[24] Stuart Geman, Elie Bienenstock, and René Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, Jan. 1992.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.