Car Detection by Fusion of HOG and Causal MRF

SATISH MADHOGARIA
Fraunhofer FKIE
Wachtberg, Germany

PAUL M. BAGGENSTOSS, Member, IEEE
Naval Undersea Warfare Center
Newport, RI, USA

MAREK SCHIKORA
WOLFGANG KOCH, Fellow, IEEE
Fraunhofer FKIE
Wachtberg, Germany

DANIEL CREMERS
Technical University of Munich
Garching, Germany

Detection of cars has a high variety of civil and military applications, e.g., transportation control, traffic monitoring, and surveillance. It forms an important aspect in the deployment of autonomous unmanned aerial systems in rescue or surveillance missions. In this paper, we present a two-stage algorithm for detecting automobiles in aerial digital images. In the first stage, a feature-based detection is performed, based on local histogram of oriented gradients and support vector machine classification. Next, a generative statistical model is used to generate a ranking for each patch. The ranking can be used as a measure of confidence or a threshold to eliminate those patches that are least likely to be an automobile. We analyze the results obtained from three different types of data sets. In various experiments, we present the performance improvement of this approach compared to a discriminative-only approach; the false alarm rate is reduced by a factor of 7 with only a 10% drop in the recall rate.

Manuscript received March 14, 2012; revised December 17, 2012, May 20, 2012, October 31, 2013; released for publication November 4, 2013.

DOI. No. 10.1109/TAES.2014.120141.

Refereeing of this contribution was handled by H. Kwon.

Authors' addresses: S. Madhogaria, M. Schikora, W. Koch, Fraunhofer FKIE, Sensor Data and Fusion, Fraunhofer Str. 20, 53343 Wachtberg, Germany, E-mail: (satish.madhogaria@fkie.fraunhofer.de, marek.schikora@fkie.fraunhofer.de, wolfgang.koch@fkie.fraunhofer.de); P. M. Baggenstoss, Naval Undersea Warfare Center, Newport, RI (paul.m.baggenstoss@ieee.org); D. Cremers, Technical University of Munich, Department of Computer Science, Informatik 9, Bolzmannstrasse 3, 85748 Garching, Germany (daniel.cremer@in.tum.de).

I. INTRODUCTION

Detecting objects in images is an interesting problem with many applications, like image retrieval and video surveillance. We have focused on detecting cars from noisy aerial images. Finding cars automatically is a hard task because the background in urban areas is highly complicated and cars tend to appear small with varying intensity or color. The appearance of the object within the observed scene also changes quite often depending on the flight altitude and camera orientation.

Most classification approaches fall into one of two categories: generative classifiers, such as density estimation methods that seek to estimate a probability density function of the data, and discriminative methods, such as support vector machines (SVMs) that seek to find the best decision boundaries. These two general approaches have been compared, with the results usually being in favor of discriminative approaches [1, 2]. Rather than choosing one method, this paper aims to make use of the advantages of both generative and discriminative classifiers. Methods based on histogram of orientation gradients (HOG) have already proved their dominance in object recognition. However, when the object is very small in a complicated background, like an aerial view of a car in images from an unmanned aerial system (UAS), edge-based detection techniques (like HOG) may result in high number of false alarms. False detections are mainly due to the close resemblance of cars to various rectangular structures in the background, such as rooftops, windowpanes, markings on the highway, and so on. In our approach, we combine the HOG-based method with a novel causal Markov random field (MRF). We have chosen to combine these two classifiers because of the complementary information that they can provide. We combine them in a pipeline approach where the first stage (HOG-SVM) aims to achieve high recall rate with a relatively high false detection rate by relaxing the decision parameters of the discriminative classifier. The second stage (causal MRF) is designed to discard most of the false positives. We demonstrate the improved performance of the two-stage algorithm over the discriminative-only approach. We also analyze the performance of the combined method on standard data sets. Finally, with our sample aerial photographs, we show that the proposed method has the potential to be used for UAS-based surveillance.

A. Related Work

Various approaches have been proposed for vehicle detection in aerial images, including a neural-network-based hierarchical model for detection [3], use of gradient features to create a generic model and Bayesian network for classification [4], feature extraction comprising geometric and radiometric features and detection using a top-down matching approach [5], and an online AdaBoost approach [6]. HOG-based features have consistently outperformed in various object detection

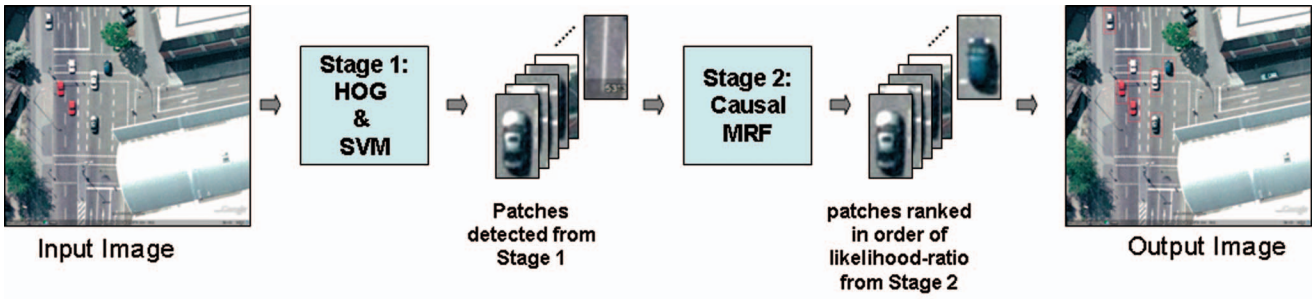

Fig. 1. Workflow of proposed two-stage algorithm for car detection.

tasks [7], however, they are limited when it comes to small objects like cars in aerial images, because many details of the cars are not always visible. There have been attempts to combine HOG features with several other feature extraction techniques for performance improvement. The most recent work is [8], where the authors combine HOG with color probability maps and pairs of pixels to form a high-dimensional feature set; this shows interesting results. We are not aware of any work which uses HOG-based detection methods in series with a generative classifier.

Some researchers have found an advantage in combining discriminative and generative classifiers [9–13]. The idea is that discriminative and generative classifiers tend to make errors for different reasons. Therefore, one can hope that a false detection made by one is likely to be rejected by the other. The cascade approach, or the detector followed by the generative classifier, is suggested for a couple reasons. First, the discriminative and generative classifiers produce different kind of outputs that are difficult to combine in a symmetric way such as addition or voting. Second, the discriminative classifier is more efficient, and thus it makes sense to eliminate a large portion of the possible noncars up front. In this work, we apply a state-of-the-art discriminative method using support vectors as a first stage, followed by a novel generative method based on a causal MRF.

B. System Overview

Fig. 1 gives an overview of the algorithm described in the paper. In Stage 1, we use the discriminative approach (HOG-SVM) to classify cars and store all detected car patches. These patches are given as an input to the MRF method (Stage 2), and a likelihood ratio value for each patch is obtained. The patches are then ranked in order of their likelihood ratios. Depending on the likelihood ratio threshold, patches are retained or rejected.

The rest of the paper is structured as follows: In Section II, we describe the discriminative approach, which makes use of histogram of orientation gradients as feature vectors and state-of-the-art support vector machines for classification. Section III describes the causal Markov random field, which uses the detected patches from the first step as inputs and either reaffirms their status as a car or rejects them. Section IV presents results and compares the outputs from the discriminative approach and the combined approach. Finally, Section V gives a short conclusion and a discussion about the possible directions for future work.

II. DISCRIMINATIVE COMPONENT

Detecting objects can be broadly divided into two steps: feature extraction and classification. Extracting features involves representing image regions as sparse or dense feature vectors. In [7], Dalal and Triggs present a human detection algorithm with excellent results. Their method uses HOG descriptors, which involve counting occurrences of gradient orientations in small regions. Subsequently, orientation histograms have been cited innumerable times for extracting feature vectors of an object or a region [6, 14–16]. HOG features seems to be a good choice for describing the shape of a car because cars have prominent edges in their appearance. Moreover, HOG has consistently been part of feature extraction techniques for vehicle detection [6, 8, 17] and has been considered an effective choice in differentiating car features from the background. HOG descriptors are similar to edge orientation histograms [14], scale-invariant feature transformations [18], and shape contexts [19], but differ in a way that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. Section II-A briefly describes each stage of HOG descriptors, derived to suit our requirements for car detection. We are interested in improving the recall rate in this step so that the first step can extract as many cars as possible. An SVM is used to classify the large set of feature vectors obtained from a small region in an aerial image.

A. Feature Vectors

From image feature sets, what is necessary is extraction of the most relevant features, which exhibit invariance to changes in illumination and differences in view points. Our feature sets are based on dense and overlapping encoding of HOG descriptors [7]. In the first step, a one-dimensional centered derivative mask is applied to the input image in order to obtain sharp changes in intensity values in both directions (Fig. 2a). Next,

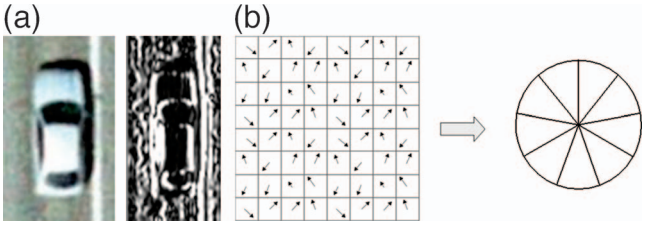

Fig. 2. (a) Gradient image in x-direction. (b) Orientations within localized portion of image (cell of size 8 × 8 pixels). All orientations within cell are put into nine-bin histogram. Any orientation within range of 0°–20° is put into first bin, 20°–40° into second bin, and so on.

orientations are computed from gradient magnitudes in both directions and accumulated for localized portions of the image, called cells, designed to reduce the illumination effects. For this purpose, the input image window is divided into cells, each of size 8 × 8. The orientations within the cell are divided into nine bins, evenly spaced over 0°–180°, so that similar orientations can be grouped together. Each pixel within the cell votes for one of the bins using the gradient magnitude information of that pixel. The histograms for each cell are contrast-normalized over a larger spatial region, called a block (Fig. 3a). We use a block size of 16 × 16 pixels. L2 normalization is used to improve performance to a great extent. Normalized vectors for overlapping blocks are combined together to form a large feature vector for one region. The parameters of HOG that produce the best result are determined through experiments. In the feature extraction process, there are mainly three variants:

1) Number of bins. As pointed out in [7], fine orientation binning is essential for good performance of the detector. Based on the recommendation in [7] and also on our own initial experiments, we conclude that nine bins is good for our application.

2) Methods of gradient computation. We use a mask [−1 0 1] to find the gradients and orientation. We experimented with a Sobel filter of kernel size 3 and 4 and could not improve the performance. Therefore, we stuck to the gradient mask proposed in [7] for gradient computation.

3) Detection window size. The size of the detection window is important, as decreasing or increasing the size affects the performance of the detector. The size is determined based on the space around the object, the shape of the object, and the ratio of the size of the object to the total size of the image. Based on our input images, we chose a 48 × 96 pixel window, in which cars of varied sizes in the same orientations can fit and at the same time leave a small border.

Besides these variants in HOG, block stride and window stride can be tuned to increase detection accuracy. In our case, a block stride of 8 × 8 is used, which means there is an overlap halfway. We keep the window stride low (8 × 8 pixels) so as to have a high recall rate in the first stage.

B. Classification

The second stage of an object detector is to decide on an appropriate classifier. Here, we went with the most modern and widely used classifier—support vector machines [20, 21]. SVMs have been applied with great success in many challenging classification problems processing large data sets. An SVM finds a hyperplane that splits the training data with maximum margin criteria. The hyperplane is characterized by a decision function

$$f(\boldsymbol{x}) = \operatorname{sgn}(\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle + b), \tag{1}$$

where $\boldsymbol{w}$ is the weight vector normal to the hyperplane, b is a bias, $b/\|\boldsymbol{w}\|$ determines the offset of the hyperplane from the origin along the normal, $\boldsymbol{x}$ is the current tested sample, and $\phi(\boldsymbol{x})$ is the kernel function. The training part of the SVM algorithm finds a $\boldsymbol{w}$ that maximizes distance between the hyperplanes. Based on the recommendations from [21] and our own initial testing, we found C-SVC with a linear kernel to reliably produce good results. Experimental results show that a nonlinear kernel tends to decrease false detection rate to a great extent but at the same time reduce recall rate. Since our intention is to maximize the recall rate in Stage 1 of the algorithm (Fig. 1), we use a linear kernel. Feature vectors from all training images are trained with SVM to build a decision model. A fixed-size sliding window is used to scan the whole image, and with the help of the decision model, a decision is made whether the computed feature vector belongs to a car or not.

For our first experiments (Google Earth data set), we keep the HOG-SVM method in its original form to prove that the combined method could actually improve the result obtained from HOG-SVM to a great extent. For a more difficult data set, we try to relax the detection in the first step so that we can have a higher recall rate. The resulting increase in false alarms is removed by the generative approach in the second step.

1) *Rotation-Invariance Detection*: As the first stage is relatively faster because of parallel implementation, during the first stage we detect cars at all possible orientations. The HOG-based model is trained using images of cars with two vertical orientations, which means the detector can only detect in two directions. To detect cars in multiple orientations, the whole image is rotated by different angles, and each time the cars are detected in the rotated image. The respective coordinates of the detected cars in the rotated image are translated back to the original image. Since HOG features already provide slight invariance in rotation (depending on the number of orientation bins), instead of smaller angles we rotate the image in steps of 30° each time, up to 150°. Note that the diagonally opposite orientations are taken care of in the training stage. Saved window coordinates and the patches representing subwindows in the input image serve as an input to the second stage.

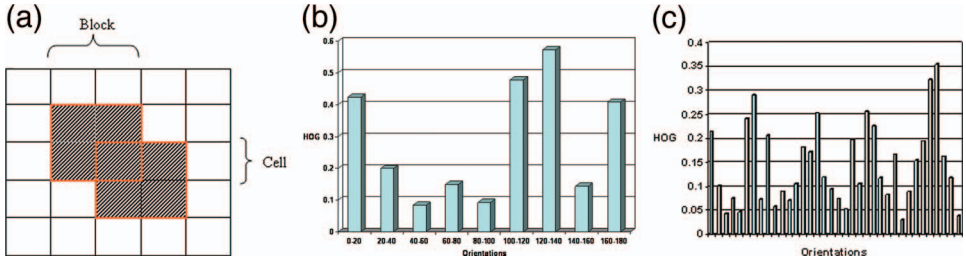

Fig. 3. In order to achieve better invariance to lighting changes, histograms of gradient orientations are normalized over larger spatial regions called blocks, as in (a). (b) Nine histogram channels for one cell (8×8). These nine values for each cell in block of size 2×2 are compiled together and normalized to obtain vector of length 36, as shown in (c).

III. GENERATIVE COMPONENT (CAUSAL MRF)

Image patches detected by the discriminative component must now be processed by the generative component to determine the likelihood ratio ranking value.

A. Motivation for Causal MRF

In Section I-A, we mentioned that advantage has been reported in combining discriminative and generative classifiers. Inspired by this work, we seek to combine HOG-SVM with a generative classifier. Since the goal of the combination is to combine the complementary characteristics of the two classifier types, it is therefore desirable to seek distinct feature types. Because HOG is based on histograms, which lose some of the spatial relationships in an image, we seek a generative model that specifically models spatial relationships. In an automobile, the spatial relationships are fairly well defined—for example, for a given orientation, the spatial relationship between the windshield, the front bumper, and the wheels is fairly constant.

A suitable model for this task may be found in the method proposed by Baggenstoss [22]. This method is a causal Markov random field operating on a two-dimensional field. It has the advantages of and MRF but the simplicity of a hidden Markov model (HMM) operating on a one-dimensional time series. By ordering the pixels in a raster-scan order, it is able to define a causal Markov relationship and arrive at a recursive algorithm modeled upon the well-known forward procedure used in the computation of the likelihood function of the HMM [23]. In addition, it applies an analog of the well-known Baum–Welch algorithm to estimate the unknown model parameters. It has been demonstrated in modeling and classifying textures in color images [22]. As described in Section III-G, we make several modifications to the algorithm to make it insensitive to color and shading and expand the pixel neighborhood to handle more complex images.

B. Mathematical Definitions

We present now the basics of the model. For additional details, the reader may refer to [22]. Let the image patch consist of a D-dimensional color vector (normally $D = 3$ for RGB color images) on a two-dimensional field:

$$\mathbf{x}_{i,j} \in \mathbb{R}^D, \quad 1 \le i \le N_1, 1 \le j \le N_2,$$

where N_1, N_2 are the pixel dimensions of the image patch. Let there be an underlying hidden discrete state at each pixel $s_{i,j}$ taking values between 1 and M. Assume that we cannot observe $s_{i,j}$ directly but instead observe a D-dimensional random vector "pixel" $\mathbf{x}_{i,j}$ whose probability density function (pdf) depends on the state $s_{i,j}$. Although in practice, image data has quantized values, we model the values as continuous-valued. The pdf is denoted by $p(\mathbf{x}_{i,j}|s_{i,j})$ or by the simpler notation $p_m(\mathbf{x}_{i,j})$ for a fixed state $s_{i,j} = m$.

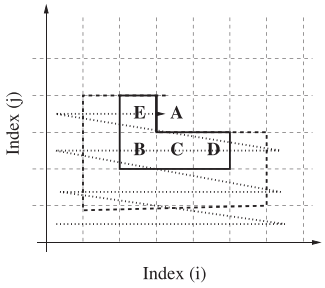

Fig. 4. Illustration of neighborhood system for pixel labeled "A" using nonsymmetric half-plane model. Probability distribution of state at pixel A depends statistically on neighbors B, C, D, and E. Zigzag arrow indicts raster-scan ordering. Expanded neighborhood is shown in dotted line for $L = 2$ (Section III-G).

C. Nonsymmetric Half-Plane (NSHP)

The nonsymmetric half-plane (NSHP) model is based on ordering the pixels in an image by scanning in a raster pattern (Fig. 4). As we scan the plane from left to right and bottom to top, when we reach pixel A we have already visited B, C, D, and E. Because each of these pixels is in the "past," we can compute the probabilities of a given state at pixel A based statistically on these four past pixels. We can extend this idea to construct a recursive forward procedure that computes state probabilities of all pixels in turn, based always on past pixels.

We define the past data $\mathbf{X}_{\langle i,j \rangle}$ as all data occurring before $\mathbf{x}_{i,j}$, that is, either directly to the left or below and

not including $\mathbf{x}_{i,j}$. More precisely,

$$\langle i, j\rangle = \{i', j'\}: \ (i' < i \ \text{ AND } \ j' = j) \ \text{ OR } \ (j' < j).$$

We also define the past and current data $\overline{\mathbf{X}}_{i,j}$ as the union of $\mathbf{X}_{\langle i,\, j\rangle}$ and $\mathbf{x}_{i,j}$. We define $p_{i,j}(m|\mathbf{X}_{\langle i,j\rangle})$ as the a priori probability of state m at pixel i,j given all data up to but not including $\mathbf{x}_{i,j}$. Similarly, we define $p_{i,j}(m|\overline{\mathbf{X}}_{i,j})$ as the a posteriori probability of state m at pixel i,j given all data up to and including $\mathbf{x}_{i,j}$. Let the index variables r, s, t, and u index over the states of the neighbor pixels (B, C, D, and E, respectively, in Fig. 4).

D. NSHP Forward Procedure

The NSHP forward procedure, analogous to the HMM forward procedure [23], recursively computes $p_{i,j}(m|\mathbf{X}_{\langle i,\, j\rangle})$ based on past information. We can start with the identity

$$\begin{aligned} p_{i,j}(m|\mathbf{X}_{\langle i,j\rangle}) &= \sum_{r,s,t,u} p_{i,j}(m, r, s, t, u|\mathbf{X}_{\langle i,j\rangle}) \\ &= \sum_{r,s,t,u} p(r, s, t, u|\mathbf{X}_{\langle i,j\rangle}) p_{i,j}(m|r, s, t, u, \mathbf{X}_{\langle i,j\rangle}) \end{aligned} \quad (2)$$

where we have used some shorthand notation including $\sum_{r,s,t,u} \triangleq \sum_{r=1}^{M}\sum_{s=1}^{M}\sum_{t=1}^{M}\sum_{u=1}^{M}$ and written the outcomes $s_{i-1,j-1} = r$ as simply r, $s_{i,j-1} = s$ as s, and so on. The Markov property [23] means that the last term in (2) can be written independent of i,j as an $M^4 \times M$ state transition matrix (STM) $\mathbf{C}_{r,s,t,u}(m)$, thus:

$$\begin{aligned} &p_{i,j}(m|r, s, t, u, \mathbf{X}_{\langle i,j\rangle}) \\ &\quad = \sum_{r,s,t,u} p(r, s, t, u|\mathbf{X}_{\langle i,j\rangle})\mathbf{C}_{r,s,t,u}(m). \end{aligned} \quad (3)$$

Our main simplifying approximation is to write $p(r, s, t, u; \mathbf{X}_{\langle i,\, j\rangle})$ as the product of the marginals

$$\begin{aligned} p(r, s, t, u; \mathbf{X}_{\langle i,j\rangle}) \simeq\ & p_{i-1,j-1}(r|\overline{\mathbf{X}}_{i-1,j-1}) p_{i,j-1}(s|\overline{\mathbf{X}}_{i,j-1}) \\ & \cdots p_{i-1,j}(u|\overline{\mathbf{X}}_{i-1,j}). \end{aligned} \quad (4)$$

This assumption treats the neighboring state outcomes as independent events, which they are obviously not, but makes the problem tractable. This type of simplifying assumption is often made in statistics, as for example the widespread use of the diagonal covariance matrix for the Gaussian model despite the fact that statistics are known to be correlated. Substituting into (2) gives us the recursion

$$\begin{aligned} &p_{i,j}(m|\mathbf{X}_{\langle i,j\rangle}) \\ &\simeq \sum_{r,s,t,u} \alpha_{i-1,j-1}(r)\alpha_{i,j-1}(s)\alpha_{i+1,j-1}(t)\alpha_{i-1,j}(u)\mathbf{C}_{r,s,t,u}(m), \end{aligned} \quad (5)$$

where we have defined the a posteriori state probabilities

$$\alpha_{i,j}(m) \triangleq p_{i,j}(m|\overline{\mathbf{X}}_{i,j}). \quad (6)$$

To complete the recursion, we need to obtain the a posteriori $\alpha_{i,j}(m)$ by updating with data available at pixel (i, j):

$$\alpha_{i,j}(m) = \frac{p_{i,j}(m|\mathbf{X}_{\langle i,j\rangle}) p_m(\mathbf{x}_{i,j})}{\sum_{n=1}^{M} p_{i,j}(n|\mathbf{X}_{\langle i,j\rangle}) p_n(\mathbf{x}_{i,j})}, \quad (7)$$

where $p_m(\mathbf{x})$, $m = 1,\ldots, M$, are the state pdfs of the continuous-valued measurements.

The joint pdf of the data up to and including $\mathbf{x}_{i,j}$ can be recursively computed in parallel using terms found in (7):

$$\begin{aligned} p(\overline{\mathbf{X}}_{i,j}) &= p(\overline{\mathbf{X}}_{i-1,j}) p(\mathbf{x}_{i,j}|\overline{\mathbf{X}}_{i-1,j}) \\ &= p(\overline{\mathbf{X}}_{i-1,j}) p(\mathbf{x}_{i,j}|\mathbf{X}_{\langle i,j\rangle}), \\ &= p(\overline{\mathbf{X}}_{i-1,j}) \sum_{m=1}^{M} p_{i,j}(m|\mathbf{X}_{\langle i,j\rangle}) p_m(\mathbf{x}_{i,j}) \end{aligned} \quad (8)$$

where the first line is Bayes's rule and the second follows, since the past and present data at $(i-1, j)$ are the same as the past data at (i, j). When the algorithm reaches the last pixel $(i = N_1, j = N_2)$, the joint forward pdf of the entire field is given by

$$L^f(\mathbf{X}) = \log p(\overline{\mathbf{X}}_{N_1,N_2}). \quad (9)$$

A very important pair of observations is that although (4) is approximate, 1) all the quantities on the right-hand side depend upon data that are strictly in the past, and 2) it is a valid probability, i.e., it sums to exactly 1 over m. The resulting data probability (9) is a pdf, so integrates to 1 over $\mathbf{X}$.

1) *Forward Procedure Initialization*: The forward procedure in (5), (7), and (8) assumes that the $\alpha_{i,j}(m)$ at all four neighbors have already been calculated. When starting up the algorithm at the first pixel $(i = 1, j = 1)$, or during the recursion at an image boundary, special handling of the past probabilities is needed. This requires defining initial state probabilities $\pi(m)$ and boundary STMs $A_u(m)$ and $\mathbf{B}_{r,s,t}(m)$, as explained in [22].

E. NSHP Backward Procedure

The forward procedure computes the forward probability of the entire field given the model. The time-reversed version, called the backward procedure [23], computes the backward probability of the entire field given the model, and is needed as a component of the Baum–Welch algorithm to re-estimate the model parameters [23]. We define

$$\beta_{i,j}(m) = p_{i,j}(m|\mathbf{X}_{\{i,j\}}), \quad (10)$$

where we define the future data $\mathbf{X}_{\{i,\, j\}}$ as all data occurring after $\mathbf{x}_{i,j}$, that is, either directly to the right or above and not including $\mathbf{x}_{i,j}$. More precisely,

$$\{i, j\} = \{i', j'\}: (i' > i \ \text{ AND } \ j' = j) \ \text{ OR } \ (j' > j).$$

Note that unlike $\alpha_{i,j}(m)$, $\beta_{i,j}(m)$ does not include information at pixel i,j. We also need to define the

data-updated version as

$$\beta'_{i,j}(m) = p_{i,j}(m|\overline{\mathbf{X}}_{\{i,j\}}) = \frac{\beta_{i,j}(m)p_m(\mathbf{x}_{i,j})}{\sum_{n=1}^{M}\beta_{i,j}(n)p_n(\mathbf{x}_{i,j})}, \quad (11)$$

where $\overline{\mathbf{X}}_{\{i,j\}}$ is the future data inclusive of $\mathbf{x}_{i,j}$. This is more analogous to $\alpha_{i,j}(m)$, since it includes $\mathbf{x}_{i,j}$. The backward recursion, the 180° rotated analog to (5), is given by

$$\beta_{i,j}(m) = \sum_{r,s,t,u}\beta'_{i+1,j+1}(r)\beta'_{i,j+1}(s)\beta'_{i-1,j+1}(t)\beta'_{i-1,j}(u)\mathbf{C}^b_{r,s,t,u}(m), \quad (12)$$

where $\mathbf{C}^b_{r,s,t,u}(m)$ is the backward version of $\mathbf{C}_{r,s,t,u}(m)$. In the first order backward procedure (i.e., HMM), it is simply the transpose of the forward STM. For the NSHP backward procedure, which is a fourth order Markov process (since it depends on the states of four past neighboring pixels), the STM is nonsquare. We have two choices. We can use the same STM in the forward and backward procedures, resulting in a symmetric algorithm insensitive to rotation by 180°, or we can estimate $\mathbf{C}^b_{r,s,t,u}(m)$ separately, resulting in a nonsymmetric algorithm. We choose the nonsymmetric algorithm for automobiles because they are not symmetric with respect to 180° rotation or time reversal in the context of NSHP ordering. Additional discussion of this topic and more details about the backward procedure may be found in [22].

The joint backward likelihood $L^b(\mathbf{X})$ is calculated analogously to (8) and (9), but time-reversed, and can be averaged with $L^f(\mathbf{X})$ to obtain a forward-backward likelihood value.

F. Overview of NSHP Baum–Welch Algorithm

We now describe an algorithm to update the NSHP model parameters analogous to the well-known Baum–Welch algorithm for the HMM [23].

1) *Gamma Probabilities*: The state probabilities given all the data are denoted by γ in HMM terminology [23]. Let

$$\gamma_{i,j}(m) = p_{i,j}(m|\mathbf{X}),$$

where $\mathbf{X}$ is all the data. We have

$$\gamma_{i,j}(m) = \frac{\alpha_{i,j}(m)\beta_{i,j}(m)}{\sum_k \alpha_{i,j}(k)\beta_{i,j}(k)}.$$

We can use γ to detect likely instances of certain states in the data. We can also use it to re-estimate state transition probabilities $\mathbf{C}_{r,s,t,u}(m)$ as well as the state likelihood functions $p_m(\mathbf{x}_{i,j})$.

2) *STM Re-Estimation*: We can re-estimate $\mathbf{C}_{r,s,t,u}(m)$ using

$$\widehat{\mathbf{C}}_{r,s,t,u}(m) = \sum_{i,j}\gamma_{i,j}(m)\gamma_{i-1,j-1}(r)\gamma_{i,j-1}(s)\gamma_{i-1,j-1}(t)\gamma_{i-1,j}(u), \quad (13)$$

followed by a normalization operation

$$\mathbf{C}_{r,s,t,u}(m) = \frac{\widehat{\mathbf{C}}_{r,s,t,u}(m)}{\sum_n \widehat{\mathbf{C}}_{r,s,t,u}(n)}. \quad (14)$$

In (13), we used the shorthand notation

$$\sum_{i,j} \triangleq \sum_{i=1}^{N_1}\sum_{j=1}^{N_2}.$$

The summation is over pixels i, j of a single image. When multiple training images are available (the usual case), we simply add image index n to the summation:

$$\sum_{i,j} \rightarrow \sum_{i,j,n}.$$

The same formulas, applied on the 180° rotated data field, can be used to estimate $\mathbf{C}^b_{r,s,t,u}(m)$ for the nonsymmetric option.

3) *State Probability Estimation*: How we re-estimate the state probabilities functions $p_m(\mathbf{x})$ depends on the mathematical form of the continuous-valued pdf model used. For the independent Gaussian model,

$$p_m(\mathbf{x}) = \prod_{k=1}^{D}\left(2\pi\sigma^2_{m,k}\right)^{-1/2} e^{-\frac{(x(k)-\mu_{m,k})^2}{2\sigma^2_{m,k}}}, \quad (15)$$

where $x(k)$ is the kth element of the pixel data $\mathbf{x}$, and we need only estimate the means and variances. This is easily done using the formulas

$$\mu_{m,k} = \frac{\sum_{i,j}\gamma_{i,j}(m)x_{i,j}(m)}{\sum_{i,j}\gamma_{i,j}(m)}, \quad (16)$$

$$\sigma^2_{m,k} = \frac{\sum_{i,j}\gamma_{i,j}(m)\left[x_{i,j}(m) - \boldsymbol{\mu}_{m,k}\right]^2}{\sum_{i,j}\gamma_{i,j}(m)}. \quad (17)$$

As before, when multiple training images are available, we simply add the image index n to the summation over i,j.

G. Causal MRF Adapted for Automobile Classification

Up to now, we have discussed the basic NSHP causal MRF as previously presented in [22]. We now make important modifications to the features to adapt to the type of data at hand. The first modification greatly increases the feature dimension, and the following five procedures are devoted to reducing the dimension of the features or parameters.

1) *Feature Vector Extension*: It is well known in automatic speech recognition that the hidden Markov model, due to its formulation in terms of a small number of discrete states, is a poor model for the dynamic behavior of the features. To greatly enhance the ability of the HMM to model continuous changes in spectra, time derivatives are appended to the feature vector [24]. We adapt this idea to images by appending the color values of neighboring

pixels to the feature vector $\mathbf{x}_{i,j}$. Adding the neighboring color values imparts the same information as adding the derivatives or differences, if the value of the given pixel is known. In order to maintain causality, we append only past pixels for the forward procedure and future pixels for the backward procedure. We use all pixels within a distance of L pixels. An illustration of the expanded neighborhood is shown in Fig. 4 for $L = 2$, where 12 neighboring pixels are seen. For $L = 3$, there are 25 neighboring pixels, and for $L = 4$, there are 40 neighboring pixels. Let H_L be the number of pixels in the neighborhood including pixel i, j; thus $H_2 = 13$, $H_3 = 26$, $H_4 = 41$, and so on. We form the data into the extended feature vector $\mathbf{w}_{i,j}$ of dimension $H_L D \times 1$. Note that neighborhood regions are different for the forward and backward algorithms (180° rotated), requiring that we define separate forward and backward feature vectors $\mathbf{w}^f_{i,j}$ and $\mathbf{w}^b_{i,j}$. To implement the NSHP forward procedure at $L = 4$ and $D = 3$, it would require estimating pdfs on 123-dimensional vectors, both means and variances. This dimensionality can be prohibitively high. We therefore take the following additional steps.

2) *Color Insensitivity*: While we wish the algorithm to be insensitive to absolute color (a green automobile or a red automobile should not affect the result), we still wish it to find boundaries between different colors (a red automobile in a green field). To achieve absolute color insensitivity but differential color sensitivity, we regard the feature vectors $\mathbf{w}^f_{i,j}$ and $\mathbf{w}^b_{i,j}$, of dimension $H_L D \times 1$, as D independent single-color $H_L \times 1$ vectors. In effect, we observe the patch of neighboring pixels separately in each color (red, green, and blue) and effectively forget which color the observations came from. Not only does this produce the right kind of color insensitivity, but it has the simultaneous benefits of reducing the feature dimension by a factor of D and increasing the number of independent observations by the same factor of D. This requires defining the six forward and backward single-color feature vectors $\mathbf{w}^{f,r}_{i,j}$, $\mathbf{w}^{f,g}_{i,j}$, $\mathbf{w}^{f,b}_{i,j}$, $\mathbf{w}^{b,r}_{i,j}$, $\mathbf{w}^{b,g}_{i,j}$, and $\mathbf{w}^{b,b}_{i,j}$.

The independence assumption implies that for RGB color data, the state observation pdfs are the product of the r, g, b components:

$$p_m(\mathbf{w}^f_{i,j}) = \prod_c p_m(\mathbf{w}^{f,c}_{i,j}), \tag{18}$$

where c indexes over the colors r, g, b, and similarly for $\mathbf{w}^b_{i,j}$. To use the extended feature vectors in the forward procedure, we replace $p_m(\mathbf{x}_{i,j})$ in (5) with (18). To used the extended feature vectors in the backward procedure, we replace $p_m(\mathbf{x}_{i,j})$ in (11) with $p_m(\mathbf{w}^b_{i,j})$, the backward equivalent of (18).

3) *State-Dependent Principal Component Analysis (PCA)*: The dimension reduction afforded by color insensitivity is good, but the dimension is still too high. To get the feature dimension to a manageable number, one could use PCA of the $H_L \times 1$ single-color extended vectors. This would involve projecting the extended feature vector $\mathbf{w}$ (we drop the subscript i, j and superscripts f, b and r, g, b to make a general expression) down to dimension P using $\mathbf{z} = \mathbf{U}'\mathbf{w}$, where $\mathbf{U}$ is an $H_L \times P$ matrix of orthonormal basis functions. In the forward procedure, we would replace $p_m(\mathbf{x})$ in (5) with $p_m(\mathbf{z})$ (we again drop subscripts i, j and superscripts f, b for generality). We need to do PCA analysis separately for the sets of forward and backward feature vectors, producing $\mathbf{U}^f$ and $\mathbf{U}^b$. It is doubtful, however, that sufficient information can be represented with a manageable number of basis functions (P around 4 or 5).

We can provide more information in the features and allow a smaller feature dimension if we adopt the technique of state-dependent PCA, in which we use a different subspace analysis for each state. We therefore assume there are state-dependent basis functions $\mathbf{U}_m$ and define the $(P + 1) \times 1$ state-dependent forward feature vector for state m:

$$\mathbf{z}_m = [\mathbf{U}'_m \mathbf{w} \log \rho^m]. \tag{19}$$

The residual energy ρ^m is defined by

$$\rho^m = \mathbf{w}'\mathbf{w} - \mathbf{w}'\mathbf{U}_m\mathbf{U}'_m\mathbf{w}. \tag{20}$$

The notation becomes cumbersome when we include all the required superscripts and subscripts:

$$\mathbf{z}^{d,c}_{m,i,j} = [\mathbf{U}^{d'}_m \mathbf{w}^{d,c}_{i,j} \log \rho^{m,d,c}_{i,j}], \tag{21}$$

where $d \in \{f, b\}$ is the direction (indicates forward or backward versions), $c \in \{r, g, b\}$ is the color, and $m \in [1, M]$ is the state. State-dependent PCA matrices $\mathbf{U}_m$ are obtained by PCA analysis of vectors $\mathbf{w}$ that have been classified according to state m. We discuss how the data are divided up according to state under the second step in Section III-H.

State-dependent PCA can be viewed as a state-dependent two-dimensional filtering that channels energy in the image corresponding to each state m into separate feature vectors. State-dependent PCA makes sense from an intuitive standpoint. For example, if a given state represents a windshield of an automobile, it is reasonable to assume that a more compact subspace (lower dimension) would be seen by PCA of just those pixel neighborhoods that were positioned on windshields. As a result of using state-dependent PCA, lower dimensional feature vectors can be used and at the same time more information can be extracted by the combined feature vectors of all states. State-dependent PCA creates a new problem, however, since it requires likelihood comparison using different feature vectors. This problem is discussed next.

4) *Pdf Projection Theorem (PPT)*: Let $\mathbf{z}_m$ be the general state-dependent PCA feature for state m and $\mathbf{z}_n$ be the state-dependent PCA feature for state n, $m \neq n$. It makes no sense to make the likelihood test

$$\log p_m(\mathbf{z}_m) \overset{<}{>} \log p_n(\mathbf{z}_n), \tag{22}$$

since these are likelihood functions defined on different features.[1] We solve this problem by applying the pdf projection theorem (PPT) [25, 26]. In general, the J-function [25] is needed when state- or class-dependent feature transformations from a common high-dimensional feature $\mathbf{w}$ to different low-dimensional features $\mathbf{z}_m$ are used in a classifier. The J-function is defined by

$$J(\mathbf{w}, \mathbf{U}_m, H_0) = \frac{p(\mathbf{w}|H_0)}{p(\mathbf{z}_m|H_0)}, \tag{23}$$

where H_0 is any statistical reference hypothesis under which we know the distribution of both $\mathbf{w}$ and $\mathbf{z}_m$. We use the reference hypothesis H_0 of independent Gaussian noise. Using the J-function, we can write the pdf of $\mathbf{w}$:

$$\log p_m(\mathbf{w}) = \log J(\mathbf{w}, \mathbf{U}_m, H_0) + \log p_m(\mathbf{z}_m). \tag{24}$$

The invalid likelihood comparison (22) is replaced by the valid likelihood comparison

$$\begin{aligned} &\log J(\mathbf{w}, \mathbf{U}_m, H_0) + \log p_m(\mathbf{z}_m) \\ &\quad \gtrless \log J(\mathbf{w}, \mathbf{U}_n, H_0) + \log p_n(\mathbf{z}_n), \end{aligned} \tag{25}$$

which is valid because the comparison is on the common feature space of $\mathbf{w}$.

The J-function for the PCA dimension reduction (21) is easily obtained when H_0 is the hypothesis that $\mathbf{w}$ is a set of independent zero-mean Gaussian samples of variance 1. Under this condition, the numerator of (23) is

$$p(\mathbf{w}|H_0) = \prod_{i=1}^{H_L} (2\pi)^{-1/2} e^{-w_i^2/2}. \tag{26}$$

Furthermore, since $\mathbf{U}_m$ is orthonormal, the first P elements of $\mathbf{z}_m$ are standard normal and independent of the residual ρ and we have

$$p(\mathbf{z}_m|H_0) = p(\log \rho|H_0) \prod_{i=1}^{P} (2\pi)^{-1/2} e^{-z_i^2/2}. \tag{27}$$

To obtain the first term, we can use the identity $p(\log \rho|H_0) = \rho p(\rho|H_0)$ and the fact that ρ is a standard chi-squared random variable with $k = H_L - P$ degrees of freedom; thus,

$$p(\rho|H_0) = \frac{2^{-k/2}\rho^{k/2-1}e^{-\rho/2}}{\Gamma(k/2)}. \tag{28}$$

Naturally, in practice, these calculations are carried out in the log domain:

$$\log J(\mathbf{w}, \mathbf{U}_m, H_0) = \log p(\mathbf{w}|H_0) - \log p(\mathbf{z}_m|H_0). \tag{29}$$

5) *Shading Insensitivity*: Shading insensitivity means we would like an algorithm that responds similarly across different illuminations (e.g., a dark automobile on a light background or a light automobile on a dark background, or illumination from various angles). Since the extended feature vector is a linear function of the image intensity, it can become inverted by inversion of the illumination. To make the algorithm insensitive to illumination and at the same time reduce the dimensionality of the parameters to estimate, we need to make it insensitive to polarity, so we assume independent zero-mean Gaussian distributions; thus,

$$p_m(\mathbf{z}_m) = (2\pi)^{-1/2} \prod_{p=1}^{P} e^{-z_m^2(p)/(2\sigma_{m,p}^2)}, \tag{30}$$

where $z_m(p)$ is the pth component of $\mathbf{z}_m$. Thus, we need only to estimate the variances. It may seem like the assumption of zero mean will remove the ability of the features to discriminate. However, since we are using state-dependent PCA, the discrimination occurs as a result of energy present in $\mathbf{z}_m$ without regard to the polarity. We will demonstrate this in Section III-J.

H. Summary of Causal MRF Algorithm

The algorithm to estimate NSHP parameters from training data and evaluate the likelihood function is summarized here.

1) **Initialize discrete probabilities.** The estimates of the a priori state probabilities $\pi(m)$ and the STMs $A_u(m)$, $\mathbf{B}_{r,s,t}(m)$, and $\mathbf{C}_{r,s,t,u}(m)$ are initialized to flat distributions (equal to the constant $1/M$).

2) **Initialize state-dependent quantities.** To obtain a starting point for estimates of state-dependent quantities such as $\mathbf{U}_m$ and the parameters of the pdfs $p_m(\mathbf{z}_m)$, we need to arbitrarily divide the data up according to state. To do this, it is necessary to cluster the data into M states using a randomly initialized clustering algorithm. We do this as follows. We obtain the $H_L \times 1$ single-color extended feature vectors as described in Section III-G-2. The forward and backward single-color extended feature vectors are concatenated as

$$\mathbf{w}_{i,j}^c = \begin{bmatrix} \mathbf{w}_{i,j}^{f,c} \\ \\ \mathbf{w}_{i,j}^{b,c} \end{bmatrix}, \tag{31}$$

where c is the color. These vectors are collected for all i,j and for all training images into the $2H_L \times N$ matrix $\mathbf{W}$, where N is the total number of vectors collected. The columns of $\mathbf{W}$ are then clustered using K-means or any general-purpose clustering algorithm into M classes. It is important that the clustering algorithm produce different clusterings over different independent trials. Once the clusters are formed, $\mathbf{W}$ is separated into the forward and backward components $\mathbf{W}^f$ and $\mathbf{W}^b$ so that PCA can be used to obtain the $H_L \times P$ state-dependent PCA subspace matrices $\mathbf{U}_m^d$. Then, state-dependent feature vectors are created according to (21) and the variances $\sigma_{m,d,p}^2$ in (30) are estimated.

[1] Here and elsewhere, we use a shorthand notation for pdfs. When we write $p(\mathbf{z})$ or $p(\mathbf{w})$, it is understood that the function $p(\ )$ changes in meaning depending on the argument, being always the pdf for the corresponding random variable argument. For different pdfs of the same argument, we use subscripts or statistical conditioning interchangeably; for example, $p_m(\mathbf{x}_{i,j}) \triangleq p(\mathbf{x}_{i,j}|s_{i,j} = m)$.

3) **Evaluate state likelihood functions.** To calculate the forward and backward procedures, we first need to evaluate the extended feature vector state log-likelihood functions $\log p_m(\mathbf{w}_{i,j}^f)$ and $\log p_m(\mathbf{w}_{i,j}^b)$, as discussed in Section III-G-2. These are calculated from feature likelihood values using (24). The feature likelihoods $p_m(\mathbf{z}_m^{f,c})$ and $p_m(\mathbf{z}_m^{b,c})$ are calculated using (30) from the variances $\sigma_{m,f,p}^2$ and $\sigma_{m,b,p}^2$.

4) **Perform forward procedure.** The forward procedure (Section III-D) is used to estimate $\alpha_{i,j}(m)$ for each pixel i,j and each state $1 \le m \le M$. Note that we replace $\mathbf{x}_{i,j}$ with the forward extended feature vectors $\mathbf{w}_{i,j}^f$, as explained in Section III-G-2. The total forward log likelihood (9) becomes $L^f(\mathbf{W}^f)$, where $\mathbf{W}^f$ is the collection of all single-color forward extended feature vectors.

5) **Perform backward procedure.** The backward procedure (Section III-E) is used to estimate $\beta_{i,j}(m)$ for each pixel i,j and each state $1 \le m \le M$. Similar to the forward procedure, we replace $\mathbf{x}_{i,j}$ with the backward extended feature vectors $\mathbf{w}_{i,j}^b$, as explained in Section III-G-2. The backward version of (8) produces the total backward log likelihood $L^b(\mathbf{W}^b)$, where $\mathbf{W}^b$ is the collection of all single-color backward extended feature vectors.

6) **Average forward and backward.** We take the average of $L^f(\mathbf{W}^f)$ and $L^b(\mathbf{W}^b)$ to obtain the forward-backward log-likelihood function $L(\mathbf{W}^f, \mathbf{W}^b)$.

7) **Find a posteriori state probabilities.** To obtain $\gamma_{i,j}(m)$ for each pixel i,j and each state $1 \le m \le M$, multiply $\alpha_{i,j}(m)$ and $\beta_{i,j}(m)$ and normalize so the quantity sums to 1 over m.

8) **Re-estimate STMs.** Use (13) followed by the normalization step to estimate $\mathbf{C}_{r,s,t,u}(m)$. If using the NSHP nonsymmetric algorithm, reorder the data in reverse (starting from top right and scanning from right to left) and use the reordered data to estimate $\mathbf{C}_{r,s,t,u}^b(m)$.

9) **Re-estimate state-dependent PCA basis.** The weighted PCA is obtained by first estimating the γ-weighted covariance

$$\mathbf{R}_m^f = \sum_i \sum_j \gamma_{i,j}(m) \sum_c \mathbf{w}_{i,j}^{f,c} \left(\mathbf{w}_{i,j}^{f,c}\right)', \qquad (32)$$

where index c runs over the three colors r, g, b, then computing the eigenvectors of $\mathbf{R}_m^f$ to produce the state-dependent PCA basis $\mathbf{U}_m^f$, and similarly for $\mathbf{U}_m^b$.

10) **Re-estimate state pdfs.** The variances $\sigma_{m,d,p}^2$—where m is the state index, d is the direction, and $1 \le p \le P$ is the PCA basis function index—are the eigenvalues of $\mathbf{R}_m^d$ in the previous step.

11) **Repeat.** Repeat Steps 3–10 until $L(\mathbf{W}^f, \mathbf{W}^b)$ ceases to increase.

12) **Perform multiple trials.** Repeat Steps 1–11 numerous times, then select the parameters that achieved the highest $L(\mathbf{W}^f, \mathbf{W}^b)$.

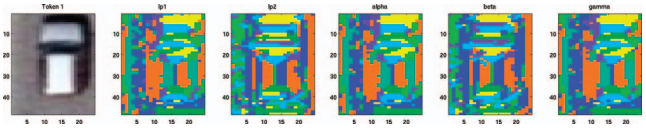

Fig. 5. Example of state estimation on training images. These images are generated by assigning each state (eight states in total) to an arbitrary color and displaying most likely state at each pixel. First image on left is patch; extreme right is most likely state; others represent intermediate stages. From left to right: (a) image patch itself; (b) most likely state based on $p(\mathbf{w}_{i,j}^f)$; (c) most likely state based on $p(\mathbf{w}_{i,j}^b)$; (d) most likely state based on $\alpha_{i,j}(m)$; (e) most likely state based on $\beta_{i,j}(m)$; (f) most likely state based on $\gamma_{i,j}(m)$.

I. Causal MRF Training and Convergence

The causal Markov random field is trained on both cars and noncars, obtaining two separate parameter sets. The training data are the same as those used for HOG; however, for causal MRF, the image patches were 2:1 down-sampled, resulting in vertically aligned automobiles occupying a region of about 12×33 pixels within image patches of size 24×48 pixels. It is interesting to observe the state estimation process at work by creating state maps, false-color images where each pixel is assigned to one of M colors (state index). Because the algorithm is initialized by randomly assigning pixels to states using an unsupervised clustering algorithm (Step 2 of Section III-H), the color (state index) to which a pixel is assigned has no physical meaning. It should, however, be consistent from patch to patch. There are a number of ways to classify a pixel into one of M states. The most logical way is assign pixel i,j to state $\hat{m} = \arg\max_m \gamma_{i,j}(m)$, since it takes into account all the data in the patch. But it is also interesting to monitor the inner workings of the algorithm by using intermediate quantities such as the forward feature likelihood $p_m(\mathbf{w}_{i,j}^f)$, the backward feature likelihood $p_m(\mathbf{w}_{i,j}^b)$, the forward a posteriori probabilities $\alpha_{i,j}(m)$, or the backward a priori probabilities $\beta_{i,j}(m)$. Fig. 5 shows one example of state estimation, with parameter values $L = 4$, $P = 5$, and $M = 8$ displaying state maps.

When observing such state maps over many training examples, one can see a consistent pattern emerging. For example, certain groups of states consistently appear on certain parts of the car. This indicates that the algorithm has learned to identify components of the automobile. The components that it identifies are unsupervised, so we must infer the meaning by observing the state maps.

The patch log likelihood $L(\mathbf{W}^f, \mathbf{W}^b)$ is computed and summed over all patches to obtain the total log likelihood. Because the algorithm is not guaranteed to increase monotonically, it typically increases very rapidly in the first five or 10 iterations, then slows and sometimes even begins to decrease. Our approach was to iterate for a maximum of 100 iterations and monitor the total log-likelihood value. The iterations were stopped once the likelihood function decreased. We used over 30 random initial starting points (different state clusterings; see

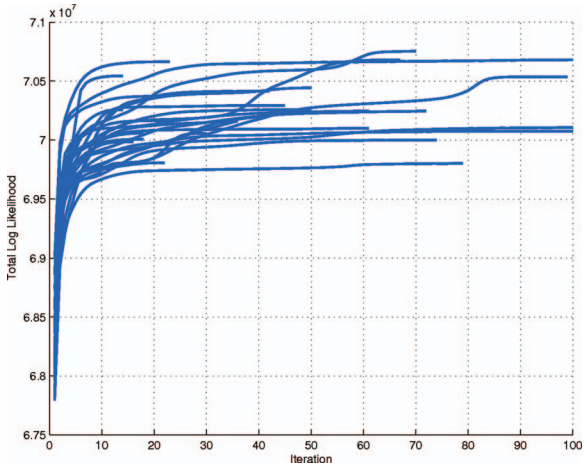

Fig. 6. Total likelihood ($L(\mathbf{W}^f, \mathbf{W}^b)$ summed over all training patches) as function of iteration number for 39 random initializations.

Section II), selecting the one that reached the highest final total log-likelihood value. Fig. 6 shows the total log likelihood for the model as a function of iteration for 39 random starting points.

J. Selection of Model Parameters

The parameters L, P, and M are important model parameters. We selected model parameters by conducting classification experiments with just the generative classifier using different parameter values. Running trials was expensive computationally, so we were only able to use three data separation experiments by dividing the data, both car and noncar, into three equal-sized sets, then training on two sets and testing on the remaining one set. The average number of correctly classified cars was determined at two fixed false alarm probabilities, 0.093 and 0.186. In Fig. 7 we plot the fraction of cars correctly classified as a function of the model parameters. At the left, we see that with $P = 4$ and $M = 7$ and L ranging from 2 to 4, the performance continually increases. Since the rate of increase slows from $L = 3$ to $L = 4$, and increasing L is computationally expensive, no experiments over $L = 4$ were conducted. In the middle, we see that with $L = 4$ and $M = 10$ and P ranging from 3 to 5, we see a peak at $P = 4$. As P becomes larger, the discrimination ability of state-dependent PCA is reduced because with larger subspaces, the state selectivity is reduced. In Section III-G-3, we mentioned that state-dependent PCA was analogous to filtering. Subspace dimension P can be likened to filter bandwidth. As the bandwidth becomes too narrow, useful energy in each band is filtered out; but as the bandwidth becomes too wide, selectivity is reduced. On the right, we see that with $L = 4$ and $P = 4$ and M ranging from 6 to 10, the general trend is increasing performance. The computational load increases at the rate of M^4, so M should be kept as low as possible. It appears that $M = 7$ or $M = 8$ is a good compromise.

K. Ranking

The patches are ranked based on their log-likelihood ratio,

$$R = L(\mathbf{W}^f, \mathbf{W}^b|H_1) - L(\mathbf{W}^f, \mathbf{W}^b|H_0), \qquad (33)$$

where H_1 and H_0 are the car and noncar hypotheses. This log-likelihood ratio is thresholded in order to reject or retain patches from the first stage.

IV. RESULTS

Since we could not find any training data available for aerial car detection online, it was prepared manually with the help of Google Earth images. We cut the top view of cars from images in Google Earth at a view height of about 100 to 120 m and rotated them into vertical positions. All car images were cut to fit in a 48×96 window size. Similarly, noncar training images ere extracted at a similar view height. We used about 200 positive samples and 700 negative samples for preparing the decision model. The number is less compared to what was used for people detection in [7], but the results obtained are fairly good. This could be due to the acts that people show more variations and the training data in [7] consisted of people in all different poses. However, in our case, we restricted cars to two vertical poses (for training), and rotation invariance is carried out during the detection stage.

A. Google Earth Data Set

First, we verified our results with images taken from Google Earth to establish the performance of the detector. The data set consisted of 30 satellite images with varying urban backgrounds and multiple cars present in each image. Typical training samples looked like the one shown in Fig. 8. For each test image, we first applied the discriminative method and stored all the detected patches. The stored patches which were identified as cars were again passed through the generative MRF test to reduce the false alarm rate. Fig. 9 shows three test examples with the precision and recall rate values. While the left column of the figure shows output from the discriminative-only approach, the right column shows outputs from the combined approach for the same images. In the case of the discriminative approach, 92% of the total cars are detected at the cost of a false alarm rate of about 37%. In the causal MRF method, the threshold value for the log-likelihood ratio could be varied to give more intensive analysis. We used different values for the threshold η, which decides whether to retain an image patch I_p or reject it:

$$I_p = \begin{cases} \text{car} & \text{if } R \le \eta \\ \text{nocar} & \text{otherwise} \end{cases}, \qquad (34)$$

where R is the log-likelihood ratio obtained from (33). In Fig. 10, we can see the effect of decreasing the

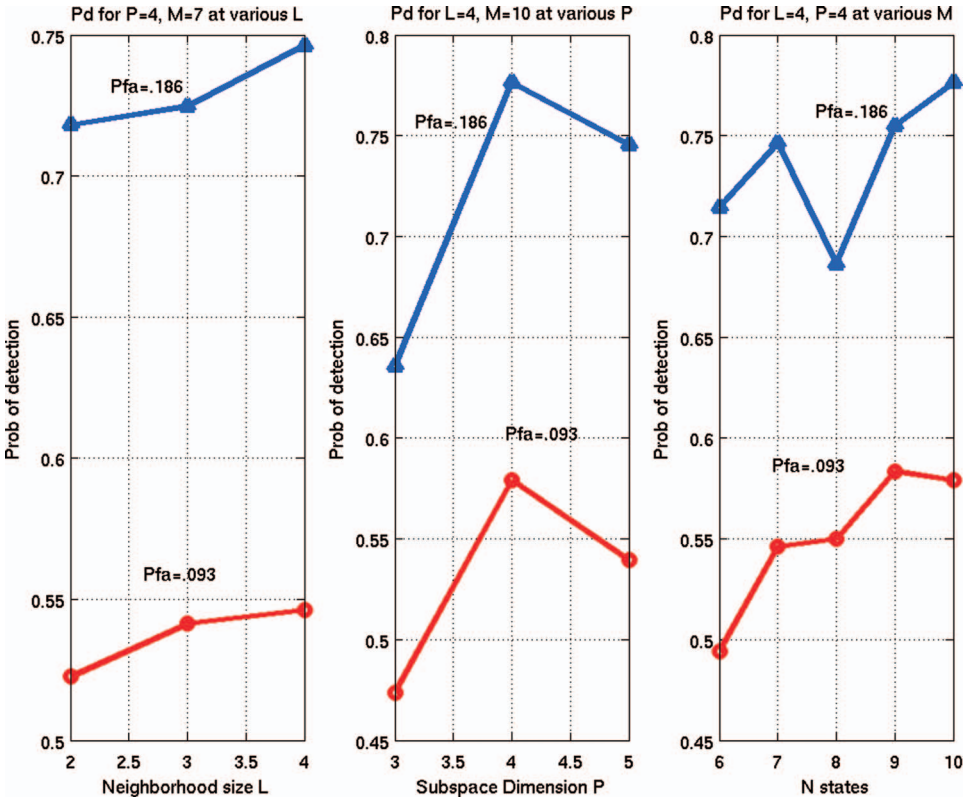

Fig. 7. Results of classification experiments to choose model parameters of generative classifier.

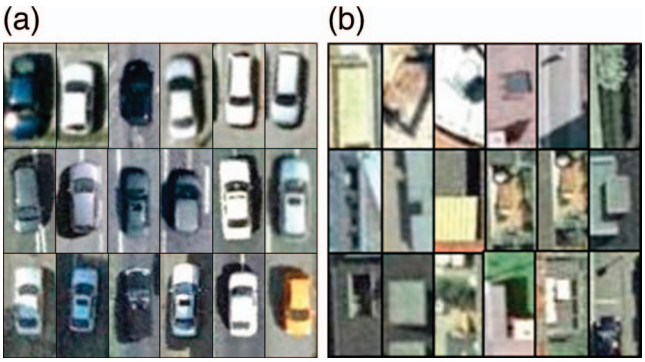

Fig. 8. Training data samples. © Google 2011.

generative-MRF threshold in the second step. Using a threshold of 3.7, we can detect all the cars from first step and still reduce the false alarm rate from 37% to 30%. On the other hand, we can reduce the false alarm rate to as low as 6% and still detect 78% of the total number of cars. With this experiment, we prove that the performance of an object detector improves remarkably by using the proposed two-stage method rather than using only HOG and SVM.

B. Overhead Imagery Research Data Set

We also tested the combined algorithm with the publicly available Overhead Imagery Research Data Set (OIRDS) [27]. We used the same SVM model as the one trained with the Google Earth data set. OIRDS is a collection of a large number of annotated overhead images. It consists of approximately 900 images, ranging in size from 257 $\times$ 257 to 500 $\times$ 500. While there are difficulties involved in terms of varying zoom level and poor quality of images, the main difficulty comes when vehicles are occluded by trees or houses. Such images are shown in Fig. 11. It would make sense to omit those images in the first place, to have a clear idea of how the combined system performs. Therefore, we tested the remaining 500 images and evaluated the performance of our detector. Since the images are taken from different altitudes, they are evaluated at multiple scales. We show some outputs from the proposed system in Fig. 12. The performance is analyzed by means of a precision recall curve shown in Fig. 13. For a clear picture of the performance, one can compare this curve with the one shown in [8] (one of the latest published results on the same data set) and deduce the superiority of our approach.

1) *Enforced High Detection in the First Stage*: Generally, while using classifiers in sequences, it is often a good practice to enforce a high detection rate in the previous stage. A high detection rate results in a very high false alarm rate too. In HOG-based detection, various parameters can be relaxed in order to improve the detection rate. For example, the threshold for the distance

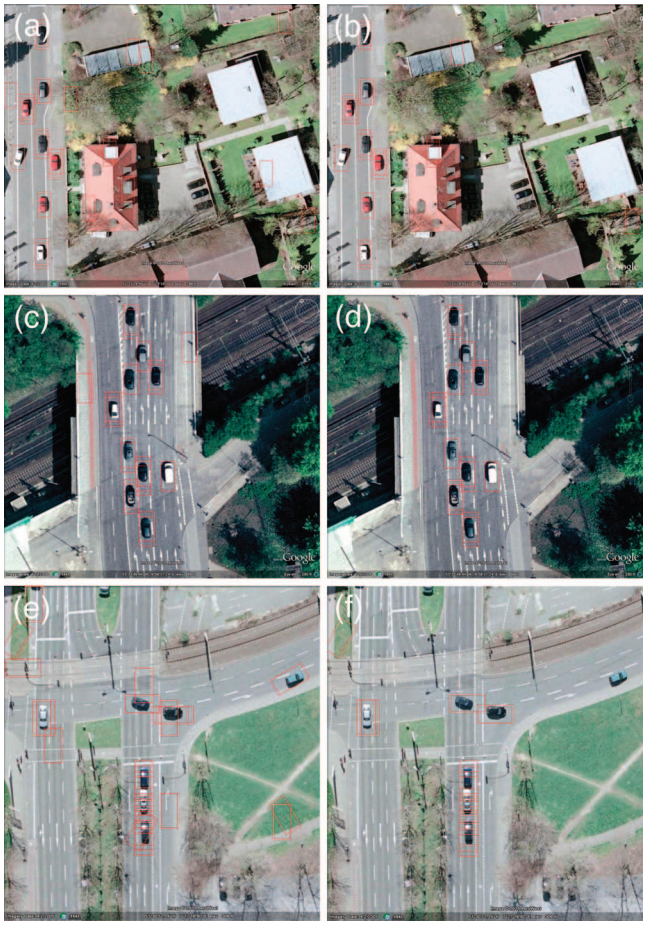

Fig. 9. Left column shows output from discriminative method, and right column from proposed generative method. We can easily see that precision rates are improved remarkably with combined method.

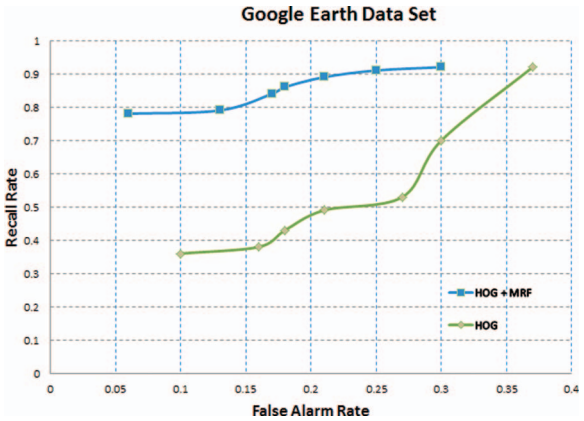

Fig. 10. Illustrates recall rate vs. false alarm for HOG and HOG + MRF methods using different classifying thresholds. Using two-stage method over HOG-only method clearly shows improvement.

between the features and the support-vector classifying plane could be lowered to increase the number of detections. On difficult data sets (OIRDS and flight images), we lowered this threshold value to maximize the detection rate in the first stage. One such example is shown in Fig. 14.

Fig. 11. Difficult images from OIRDS. In many cases, cars are difficult to find, with some of them occluded or edges not clearly visible, and so on.

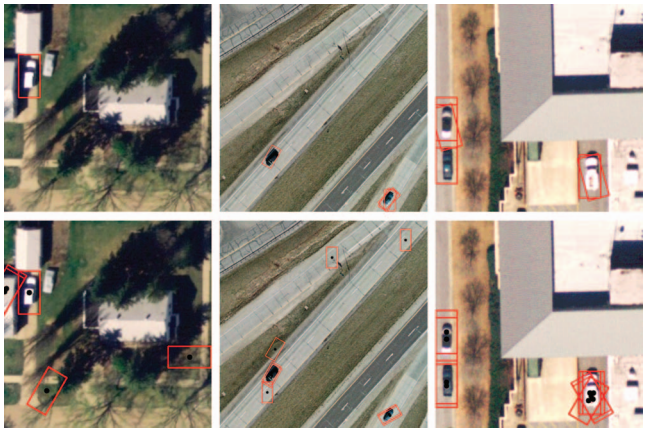

Fig. 12. OIRDS: Top row shows two-stage result, while bottom row shows HOG + SVM results.

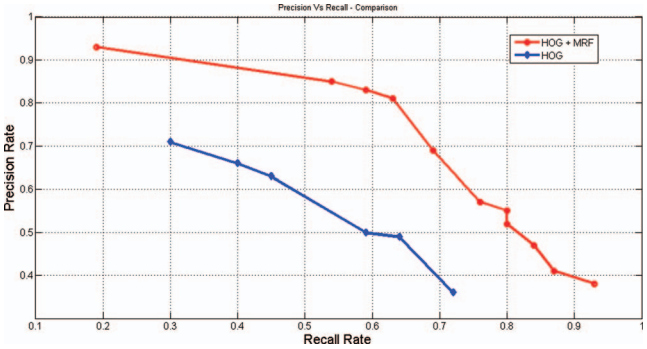

Fig. 13. Performance of two-stage algorithm on OIRDS.

C. Images from Flight

While the previous two data sets serve the purpose of establishing that the combined method works rather well, the main aim of this work would be to be able to identify objects of interest in UAS images for surveillance. We obtained a set of 15 images captured by a fixed camera mounted beneath an aircraft. These are relatively high-resolution images taken from altitudes ranging from 100 to 200 m. Each image is approximately 2500 $\times$ 2500, with very few cars in it. A large search area and small number of cars in one image automatically increases the false alarms compared to other data sets, where the search

Fig. 14. Varying thresholds in discriminative classifier. Threshold is lowered as much as possible so that it can detect maximum cars in first stage. False alarm thus also increases, but overall performance is improved with combined method.

Fig. 15. Flight images taken at different altitudes.

space is small and contains a relatively large number of cars. Overall, we could detect 70% of the cars with about 30% false detection. In Fig. 15, some of the results are displayed; the precision versus recall rate in Fig. 16 illustrates the performance of our approach on flight images. As the altitude from which the images are captured varies greatly, the scaling factor must be adjusted with respect to the height. This is due to the fact that we have a fixed size for training images, which therefore limits our application to detect fixed-size cars.

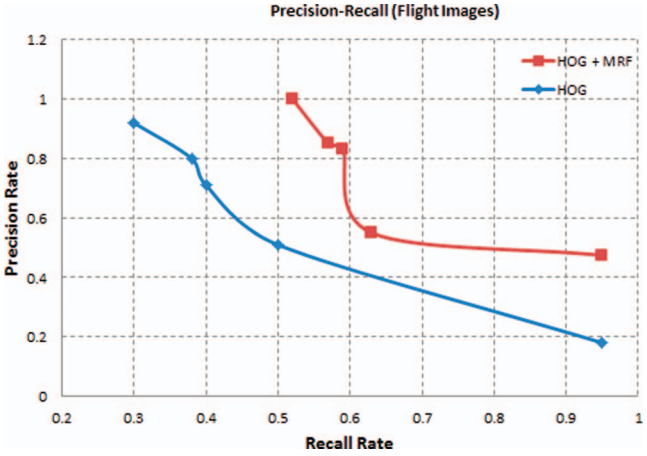

Fig. 16. Precision vs. recall rate for flight images.

V. CONCLUSION

We have presented a two-stage algorithm for detecting cars in aerial images. The main idea was to combine the benefits of a discriminative and a generative approach for object detection in order to improve the performance of either method when used individually. We tested the proposed algorithm with three different types of images. With the first data set (Google Earth), we proved how false alarm rates from standard HOG and SVM could be improved with the help of a causal MRF method. In the second experiment (OIRDS), we showed that the combined algorithm can be used in low-resolution images as well by tuning the parameters of HOG for a higher detection rate in the first stage. Finally, we tested the combined method with relatively high-resolution camera images obtained from flight and proposed the potential use of this approach for detecting small objects, i.e., cars from aerial images.

REFERENCES

[1] Rubinstein, Y. D., and Hastie, T.
Discriminative vs informative learning.
In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, CA, 1997, 49–53.

[2] Jebara, T.
Machine Learning: Discriminative and Generative.
Boston: Kluwer, 2004.

[3] Ruskoné, R., Guigues, L., Airault, S., and Jamet, O.
Vehicle detection on aerial images: A structural approach.
In *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, 1996, **3**, 900–904.

[4] Zhao, T., and Nevatia, R.
Car detection in low resolution aerial images.
In *Eighth International Conference on Computer Vision*, Vancouver, Canada, 2001.

[5] Hinz, S.
Detection and counting of cars in aerial images.
In *2003 International Conference on Image Processing*, Barcelona, Spain, 2003.

[6] Nguyen, T. T., Grabner, H., Bischof, H., and Gruber, B.
On-line boosting for car detection from aerial images.
In *2007 IEEE International Conference on Research, Innovation and Vision for the Future*, Hanoi, Vietnam, 2007.

[7] Dalal, N., and Triggs, B.
Histograms of oriented gradients for human detection.
In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005. Available: http://lear.inrialpes.fr/pubs/2005/DT05.

[8] Kembhavi, A., Harwood, D., and Davis, L. S.
Vehicle detection using partial least squares.
IEEE Transactions on Pattern Analysis and Machine Intelligence, **33**, 6 (June 2011), 1250–1265.

[9] Fine, S., Navrátil, J., and Gopinath, R. A.
Enhancing GVM scores using SVM "hints."
In *Proceedings of the Seventh European Conference on Speech Communication and Technology*, Aalborg, Denmark, 2001. Available: citeseer.ist.psu.edu/fine01enhancing.html.

[10] Jaakkola, T. S., and Haussler, D.
Exploiting generative models in discriminative classifiers.
In *Advances in Neural Information Processing Systems 11*, Denver, CO, 1998, 487–493.

[11] Quan, L., and Bengio, S.
Hybrid generative-discriminative models for speech and speaker recognition. Dalle Molle Institute for Perceptual Artificial Intelligence, IDIAP Research Rep. 02–06, Mar. 2002.

[12] Raina, R., Shen, Y., Ng, A. Y., and McCallum, A.
Classification with hybrid generative/discriminative models.
In *Advances in Neural Information Processing Systems 16*. Vancouver, Canada, 2003.

[13] Harrison, B. F., and Baggenstoss, P. M.
Hybrid discriminative/class-specific classifiers for narrowband signals.
IEEE Transactions on Aerospace and Electronic Systems, **44**, 2 (Apr. 2008), 629–642.

[14] Freeman, W. T., and Roth, M.
Orientation histograms for hand gesture recognition.
In *Proceedings of the IEEE International Workshop on Automatic Face and Gesture Recognition*, Zurich, 1994. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.618rep=rep1type=pdf.

[15] Marimon, D., and Ebrahimi, T.
Orientation histogram-based matching for region tracking.
In *Eighth International Workshop on Image Analysis for Multimedia Interactive Services*, Santorini, Greece, 2007. Available: http://mkg.iti.gr/wiamis2007.

[16] Ott, P., and Everingham, M.
Implicit color segmentation features for pedestrian and object detection.
In *IEEE 12th International Conference on Computer Vision*, Kyoto, Japan, 2009, 723–730.

[17] Gleason, J., Nefian, A. V., Bouyssounousse, X., Fong, T., and Bebis, G.
Vehicle detection from aerial imagery.
In *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, 2065–2070.

[18] Lowe, D. G.
Distinctive image features from scale-invariant keypoints.
International Journal of Computer Vision, **60**, 2 (Nov. 2004), 91–110.

[19] Belongie, S., Malik, J., and Puzicha, J.
Matching shapes.
In *Eighth IEEE International Conference on Computer Vision*, Vancouver, Canada, July 2001, **1**, 454–461.

[20] Burges, C. J. C.
A tutorial on support vector machines for pattern recognition.
Data Mining and Knowledge Discovery, **2**, 2 (1998), 121–167.

[21] Chang, C.-C., and Lin, C.-J.
LIBSVM: A library for support vector machines,

http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[22] Baggenstoss, P. M.
Two-dimensional hidden Markov model for classification of continuous-valued noisy vector fields.
IEEE Transactions on Aerospace and Electronic Systems, **47**, 2 (Apr. 2011), 1073–1080.

[23] Rabiner, L. R.
A tutorial on hidden Markov models and selected applications in speech recognition.
In *Readings in Speech Recognition*, A. Waibel and K.-F. Lee, Eds. San Francisco: Morgan Kaufmann Publishers, 1990, ch. 6.1, 267–296.

[24] Furui, S.
Speaker-independent isolated word recognition using dynamic features of speech spectrum.
IEEE Transactions on Acoustics, Speech and Signal Processing, **34**, 1 (Feb. 1986), 52–59.

[25] Baggenstoss, P. M.
The PDF projection theorem and the class-specific method.
IEEE Transactions on Signal Processing, **51**, 3 (Mar. 2003), 672–685. Available: http://ieeexplore.ieee.org/xpls/abs/all.jsp?arnumber=1179761

[26] Baggenstoss, P. M.
A modified Baum–Welch algorithm for hidden Markov models with multiple observation spaces.
IEEE International Conference on Acoustics, Speech, and Signal Processing, Istanbul, 2000, **2**, II717–II720.

[27] Tanner, F., Colder, B., Pullen, C., Heagy, D., Eppolito, M., Carlan, V., Oertel, C., and Sallee, P.
Overhead Imagery Research Data Set—An annotated data library and tools to aid in the development of computer vision algorithms.
In *2009 IEEE Applied Imagery Pattern Recognition Workshop*, Washington, DC, 2009.

Satish Madhogaria received a B.Tech. (CS) degree from West Bengal University of Technology, India, in 2008 and an M.S. degree in computer science from the University of Bonn, Germany, in 2010. Since August 2010, he has worked in the sensor data and information fusion department at Fraunhofer FKIE. His research is focused on image-based classification, object detection, and application of machine-learning techniques to real-world problems.

Paul M. Baggenstoss received his Ph.D. degree in electrical engineering (statistical signal processing) at the University of Rhode Island in 1990. He joined the Naval Undersea Warfare Center (NUWC) Newport, Rhode Island, in 1996 and remains there today. At NUWC, he has applied statistical signal processing and classification theory to problems in underwater acoustics. He has taught as an adjunct professor of electrical engineering at the University of Connecticut, Storrs, and has participated in two one-year foreign research assignments in Germany, most recently in 2010 at Fraunhofer FKIE in Bonn. He is the author of numerous conference and journal papers in the field of signal processing and classification.

Marek Schikora received a B.S. degree (Vordiplom, 2006) and M.S. degree (Diplom, 2008) in computer science from the University of Bonn, Germany. Since 2009 he has been a research scientist in the sensor data and information fusion department at Fraunhofer FKIE. In addition, he is a Ph.D. student in the Computer Vision Group at the Technical University of Munich, Germany. His research is focused on sensor fusion, multitarget tracking, and related computer vision topics, like image segmentation, object detection, and classification.

Wolfgang Koch studied physics and mathematics at RWTH Aachen. He is the head of the sensor data and information fusion department at Fraunhofer FKIE, a research institute active in defense and security. On fusion topics, he has published several handbook chapters and numerous journal/conference articles. For *IEEE Transactions on Aerospace and Electronic Systems*, he is associate editor-in-chief and technical editor. Moreover, he is a member of the board of directors of the International Society of Information Fusion. At Bonn University he holds a habilitation degree in applied computer science and gives a regular lecture series on sensor data and information fusion. He initiated the SDF: Sensor Data Fusion: Trends, Solutions, Applications series of annual IEEE workshops. In 2008 he was the executive chair of the 11th International Conference on Information Fusion, June 30–July 3, in Cologne, Germany. In 2010, he was elevated to the grade of fellow of the IEEE.

Daniel Cremers received an M.S. (Diplom) degree in theoretical physics from the University of Heidelberg, Germany, in 1997, and a Ph.D. degree in computer science from the University of Mannheim, Germany, in 2002. Subsequently, he spent two years as a postdoctoral researcher at the University of California, Los Angeles, and one year as a permanent researcher at Siemens Corporate Research, Princeton, New Jersey. From 2005 to 2009, he headed the Computer Vision Group at the University of Bonn, Germany. Since 2009, he has been a full professor at TU München. He has received several awards, in particular the Best Paper of the Year Award 2003 from the Pattern Recognition Society, the Olympus Award 2004, and the 2005 UCLA Chancellor's Award.