Vehicle Detection Using Partial Least Squares

Aniruddha Kembhavi, David Harwood, *Member*, *IEEE*, and Larry S. Davis, *Fellow*, *IEEE*

Abstract—Detecting vehicles in aerial images has a wide range of applications, from urban planning to visual surveillance. We describe a vehicle detector that improves upon previous approaches by incorporating a very large and rich set of image descriptors. A new feature set called *Color Probability Maps* is used to capture the color statistics of vehicles and their surroundings, along with the Histograms of Oriented Gradients feature and a simple yet powerful image descriptor that captures the structural characteristics of objects named *Pairs of Pixels*. The combination of these features leads to an extremely high-dimensional feature set (approximately 70,000 elements). Partial Least Squares is first used to project the data onto a much lower dimensional subspace. Then, a powerful feature selection analysis is employed to improve the performance while vastly reducing the number of features that must be calculated. We compare our system to previous approaches on two challenging data sets and show superior performance.

Index Terms—Vehicle detection, partial least squares, feature selection.

1 Introduction

Several commercial earth observation satellites, such as IKONOS, GeoEye, and QuickBird, provide publicly available imagery at a ground sampling distance (GSD) of 1 meter. High-resolution images of a small number of locations are publicly available via Google Earth at an astonishing GSD of 0.15 meter. We consider the problem of detecting vehicles from such high-resolution aerial and satellite imagery; this problem has a number of applications. Images of road networks, along with the distribution of vehicles in different regions, can provide information for urban planning and traffic monitoring. Detecting and tracking vehicles in aerial videos is also an important component in visual surveillance systems. In spite of the increasing availability of high-resolution aerial and satellite images, vehicle detection still remains a challenging problem. In urban settings especially, the presence of a large number of rectilinear structures, such as trash bins, electrical units, and air conditioning units on the tops of buildings, can cause many false alarms. Fig. 1a shows an example of an image patch for which previously published vehicle detectors produce a large number of false alarms.

Object detection systems have typically used image descriptors such as Scale Invariant Feature Transform (SIFT) [17] and Geometric Blur [2], calculated at a number of interest points within the image. These image descriptors are then combined using various aggregating approaches, such as Bags-Of-Words [32] and Spatial Pyramids [16], to provide a rich description of the object. However, such approaches have not been extensively used for the problem of vehicle detection due to the low resolution of traditional aerial images and the need for many commonly used image descriptors for a sufficiently large support region.

Vehicle detection has been previously treated as a template matching problem, and several algorithms have been proposed that construct templates in 2D as well as 3D. Moon et al. [20] propose an approach to optimally detect 2D shapes. They derive an optimal 1D step edge detector which minimizes the noise power and mean squared error between the input and filter output. This turns out to be the derivative of the double exponential (DODE) function. The DODE filter is then extended along the shape's boundary contour to obtain a shape detector. The problem of vehicle detection is then equivalent to *detecting parallelograms*. They show impressive results on images of vehicle parking lots. A comprehensive analysis of this vehicle detector under a wide range of operating environments was carried out in [21]. A mathematical analysis was provided to quantify the degradation of the vehicle detector with decreasing illumination and acquisition angle.

Choi and Yang [5] use a mean-shift-based clustering algorithm to extract candidate blobs that exhibit symmetry properties of typical vehicles. Each candidate is then classified using geometric and radiometric characteristics of the blob. Eikvil et al. [7] propose a similar two-stage strategy for vehicle detection in satellite images. The first stage consists of segmenting regions into potential vehicles, roads, vegetation, etc. They also leverage multispectral information to identify regions of vegetation and Geographical Information System (GIS) data to obtain the road network. The second stage then consists of a region classification algorithm using geometrical properties such as area, moments, etc. Zheng et al. [34] obtain vehicle candidates using a morphological preprocessing stage which are then classified using a neural network. Such two-stage approaches typically suffer from errors obtained in the segmentation stage of the system. Furthermore, geometric properties of blobs are not powerful enough to detect vehicles with high accuracy in urban settings, where the presence of a large number of rectilinear structures cause many false alarms.

- *A. Kembhavi is with Microsoft Corporation, aniruddk, City Center/16503, 1 Microsoft Way, Redmond, WA 98052. E-mail: anikem@umd.edu.*
- *D. Harwood and L.S. Davis are with the Institute for Advanced Computer Studies, University of Maryland, A.V. Williams Building, College Park, MD 20742. E-mail: harwood@umiacs.umd.edu, lsd@cs.umd.edu.*

Manuscript received 8 Dec. 2009; revised 26 Apr. 2010; accepted 10 Aug. 2010; published online 29 Sept. 2010.
Recommended for acceptance by S. Li.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-12-0805.
Digital Object Identifier no. 10.1109/TPAMI.2010.182.

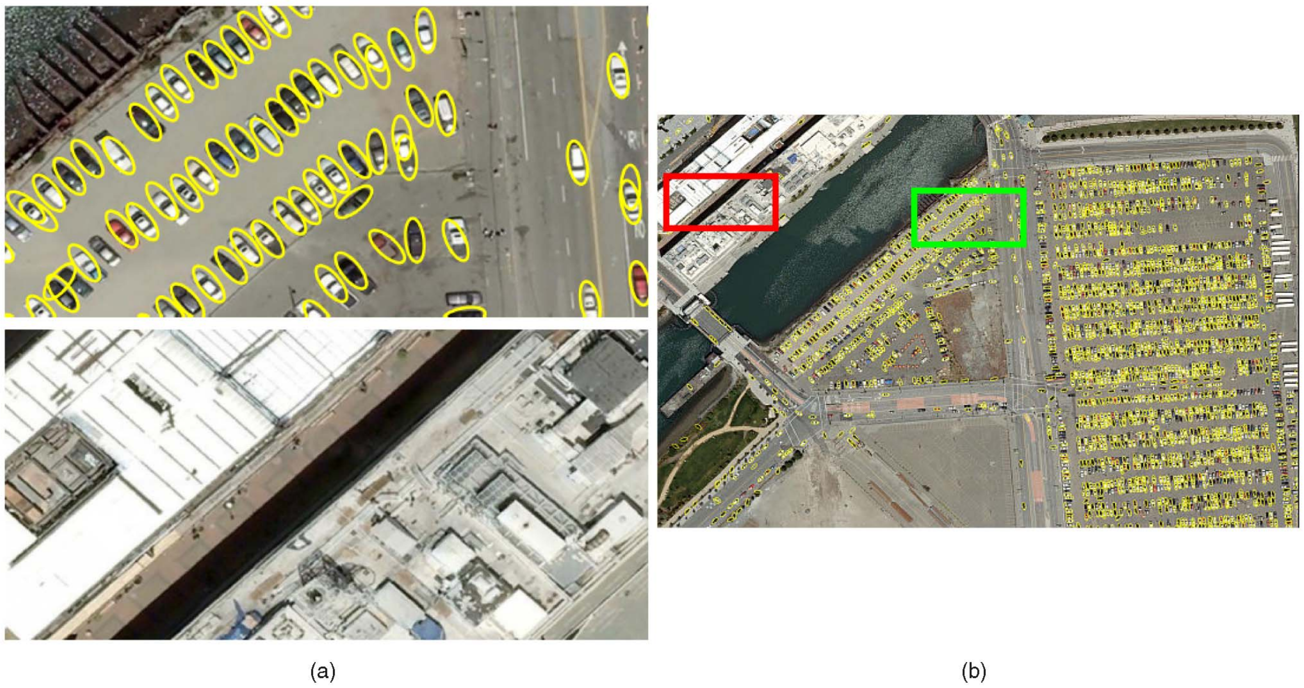

Fig. 1. (a) Two image patches showing the performance of our system. Both image patches were extracted from a much larger image ($5{,}007 \times 7{,}776$ pixels) which is displayed in its entirety in (b) and at a higher resolution in Fig. 23. The top figure (region marked in green) shows the high detection accuracy of our system in the presence of a large number of vehicles. The bottom image (region marked in red) shows the low false alarm rate of our system in a region that has many rectilinear structures. Typical vehicle detection systems often produce false alarms in the presence of such structures. © 2009 Google.

Zhao and Nevatia [33] pose the vehicle detection problem as a 3D object recognition problem. They use human knowledge to model the geometry of a typical vehicle. Psychological tests revealed that human subjects most often used cues such as the rectangular shape of the car, layout of windshields, and presence of shadows to detect cars. Such cues are then integrated using a Bayesian network. They also make use of camera calibration and illumination information to predict shadow cues. While effective, their algorithm cannot be easily extended to build other object detection systems due to the large amount of human modeling that is required. Similar 3D models have also been used to model vehicle geometries for the purpose of car detection and counting in aerial images by Hinz [15] and Schlosser et al. [25].

Grabner et al. [12] propose an online version of boosting to efficiently train their vehicle detector. Their algorithm avoids building large prelabeled training data sets by using an active learning framework. They use three classes of features—Haar wavelets, Histograms of Oriented Gradients, and Local Binary Patterns, all of which can be very efficiently calculated using Integral Images and Integral Histograms. Their detection results are further improved by segmenting the image into streets, buildings, trees, etc., and then discarding vehicle detections that are not present on the streets.

More recently, vehicle analysis has been extended from single images to video sequences. Yue et al. [31] propose a system for vehicle verification in airborne video sequences. The vehicle of interest may leave the field of view for a while or may be obscured. When a new vehicle is observed, verification is needed to confirm whether it was the previously detected vehicle. A homography-based view synthesis method is used to generate novel views of the exemplars that are provided during training. This enables the system to be robust to large aspect angle variations of the test sequence. The synthesized novel view and testing object are then compared using a weighted combination of a rotationally invariant color matcher and a spatial feature matcher.

Our proposed vehicle detector improves upon previous systems by incorporating a much larger and richer feature set than previous approaches. First, a novel set of image descriptors is proposed that capture the color properties of an object and its surrounding, called *Color Probability Maps (CPM)*. Then, the commonly used Histograms of Oriented Gradients (HOGs) feature is incorporated to capture the spatial distribution of edge orientations. Finally, a very simple yet powerful image descriptor, named *Pairs of Pixels (PoPs)*, is proposed to capture the structural properties of objects. The concatenation of these three classes of features leads to a very high-dimensional feature vector (approximately 70,000 elements). In contrast, the number of samples that we have to train our vehicle detector is much smaller, rendering many popular machine learning techniques unusable. Furthermore, our features are extracted from neighboring pixels within a detection window, which greatly increases their multicollinearity. We take advantage of the nature of our problem by employing a classical statistical regression analysis technique called Partial Least Squares (PLS). Our PLS analysis extracts a low-dimensional subspace, within which we can use a simple quadratic discriminant classifier to classify image patches into vehicles and background.

Using such a large number of features greatly increases the computational cost of the vehicle detector. A common approach to speed up detectors using a large number of features is to use a boosting algorithm along with a rejection cascade, as in [28]. We reduce the computational cost of our system using a dual-feature selection approach. First, a recently proposed feature selection method called Ordered Predictors Selection, which combines a number of informative vectors that rank features based on their predictive performance, is used. This is coupled with multistage, multiresolution image analysis, where a large number of image windows are discarded at an early stage of processing (processing at lower resolutions) and only a small fraction of image patches are analyzed at the highest image resolution (second stage). Our feature selection approach not only increases the speed of our system but also its performance.

We demonstrate our proposed vehicle detector on two data sets. The first consists of color images collected from a satellite and obtained via Google Earth. This is a set of 40 high-resolution images over the city of San Francisco. The second data set is the publicly available Overhead Imagery Research Data Set, which consists of a large number of aerial images, annotated with vehicles [26]. We compare our approach to several previously proposed object detection approaches, including the Histograms of Oriented Gradients approach of Dalal and Triggs [6], the Spatial Pyramidal Matching algorithm [16] incorporating SIFT features, the recently proposed Intersection Kernel Support Vector Machines using HOG features [19], and a vehicle detector proposed in [20]. Our solution obtains favorable results as compared to previous approaches.

2 Feature Extraction

We use three classes of features in our proposed solution: Color Probability Maps, Histograms of Oriented Gradients, and PoPs features.

2.1 Color Probability Maps

Vehicles often lie on homogeneously colored backgrounds such as asphalt, cement, dirt roads, etc. Sometimes the immediate neighborhood of a vehicle might be composed of multiple surfaces (a vehicle parked on the side of the road has an asphalt surface on three sides and a cement sidewalk on the fourth side). Thus, an image patch containing typical vehicle colors toward the center and colors representing typical backgrounds toward the periphery is likely to contain a vehicle. Color Probability Maps capture such color statistics of objects and their immediate environment.

We begin by identifying colors that are typically present in the background category. First, pixels are sampled from the entire set of background image patches in the training set of images. Each pixel is represented in a 3D space (r, g, s), where r and g are chromaticity variables representing the fraction of the red and green components, respectively: $r = \frac{R}{R+G+B}$ and $g = \frac{G}{R+G+B}$. s represents the brightness component: $s = \frac{R+G+B}{3}$. These pixels (obtained from all of the background training images) are clustered to determine the dominant colors present in the background. Each cluster of pixels is used to build a color density model in RGB space using Kernel Density Estimation:

$$\begin{aligned} p^c(r,g,s) &= \frac{1}{N^c_{pts}} \sum_{i=1}^{N^c_{pts}} K_{\sigma_r}\big(r - r^c_i\big) K_{\sigma_g}\big(g - g^c_i\big) K_{\sigma_s}\big(s - s^c_i\big), \\ K_\sigma(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}, \end{aligned} \quad (1)$$

where $p^c(r,g,s)$ refers to the probability of the (r,g,s) triple for the cth cluster. σ_r, σ_g, and σ_s represent the channel bandwidths. r^c_i, g^c_i, and s^c_i refer to the chromaticity and brightness components of the ith pixel in the cth cluster. N^c_{pts} refers to the number of points in the cth cluster. Given an image, one can obtain a color probability map for every color model. These probability maps are concatenated to form a feature vector representing the color statistics of the image patch, F_{cmap}. In order to limit the number of probability maps that must be computed for each image patch, only the most discriminating clusters are used. First, clusters that contain a very small number of points are rejected. Then, the remaining clusters are ranked based on their discriminative capability. For a given cluster, we generate the corresponding probability map for all positive and negative samples in the training set and calculate the average misclassification error using a fivefold cross-validation procedure. The top N_{cmap} clusters are then chosen. Fig. 2 shows a schematic of this feature calculation.

The Improved Fast Gauss Transform (IFGT) [22] is used for efficient computation of the probabilities. The bandwidths for each channel are estimated independently using a bandwidth selection criterion given in [23]. Given a test image patch, the computation of its Color Probability Maps is further speeded up by a priori constructing lookup tables that directly map entries in the $R - G - B$ color space to a probability value. A lookup table is constructed for each of the N_{cmap} color clusters.

2.2 Histograms of Oriented Gradients

The second class of features are the HOGs, which have been used in many object detection algorithms [6]. These features capture the spatial distribution of gradients that are typically observed in image patches that contain vehicles. Since histograms are computed over regions, they are fairly robust to some variability in the location of the parts of the object. Moreover, the HOG descriptor is also invariant to rotations smaller than the size of the histogram orientation bin.

Each detection window is divided into square cells and a 9-bin HOG feature is calculated for each cell. Grids of 2×2 cells are grouped into a block, resulting in a 36D feature vector per block. Each block feature vector is normalized to an L2 unit length. Dalal and Triggs [6] used blocks defined at a single scale. In their approach, for a detection window of size 64×128 pixels, 105 blocks were used, each having a size of 16×16 pixels. Zhu et al. [35] extended this approach by using blocks at varying scales and varying aspect ratios (1:1, 1:2, and 2:1). We incorporate this multiscale approach employed in [35].

2.3 Pairs of Pixels

Properties of pixel pairs within an image patch can provide structural information about the object present in that patch. Consider the image patch shown in Fig. 3. The structure of a car shown in the image can be described using the relationships between the regions highlighted in red, green,

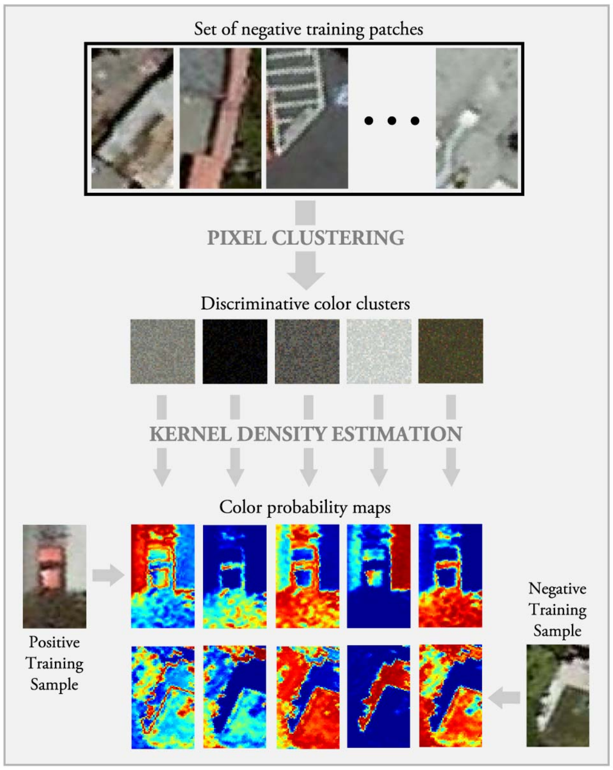
Set of negative training patches
PIXEL CLUSTERING
Discriminative color clusters
KERNEL DENSITY ESTIMATION
Color probability maps
Positive Training Sample
Negative Training Sample

Fig. 2. Color probability maps. Pixels extracted from negative training image patches are clustered to obtain models of typical colors observed in the background. Given a new image patch, kernel density estimation is then used to obtain a probability map corresponding to each color cluster. © 2009 Google.

and blue. The three regions highlighted in red represent the body of the vehicle and typically have the same color. Similarly, regions highlighted in green represent the windows of the vehicle which usually appear dark in color. Regions that are not present on the vehicle might have colors that differ from the color of the vehicle, but they might be similar to one another. Such relative color statistics can capture the structure and relationships among different parts of a given object. Using relative properties of pixel pairs, as opposed to pixel properties themselves, is robust to illumination changes in the scene, changes in the background, as well as the color of the object itself.

In principle, one can encode many different relative properties of regions, such as the difference between their colors, textural properties, gradient magnitudes, etc. Here,

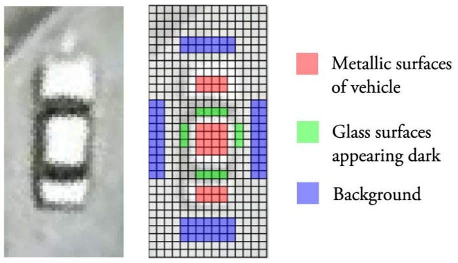

Fig. 3. The structure of a typical car and its surrounding regions can be described using pairwise relationships between the highlighted regions. Regions that are marked with the same color typically have the same color and texture properties. This information is captured by the PoP feature.

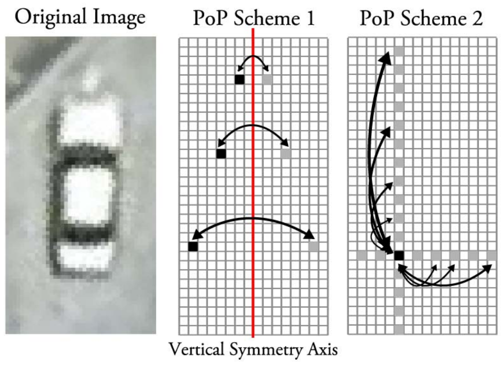

Fig. 4. POP schemes. Scheme 1 captures differences between pairs of pixels that lie symmetrically across the central vertical axis of the image patch. Scheme 2 captures differences between pairs of pixels that lie in the same row and column.

we restrict these regions to be single pixel locations and the relative property to be the euclidean distance between their color values. The feature vector F_{pop} encoding this relative property can be obtained by concatenating the distances between the colors of a large number of pixel pairs.

If we were to consider all pixel pairs in an image patch, the feature vector would be of length M^2N^2, where M and N are the length and width of an image patch. For a small image patch of size 100×100, this would result in a feature vector of length 100 million. In order to restrict this dimensionality, we propose two alternatives. These alternatives are designed to capture a large portion of the structural information while taking advantage of the symmetry exhibited by vehicles about a central vertical axis.

Scheme 1. Consider all pairs of pixels that exhibit a symmetry in location about the central vertical axis, as shown in Fig. 4. This reduces the dimensionality of the feature vector to $M \times \frac{N}{2}$. However, these pixel pairs only capture horizontal differences.

Scheme 2. Consider a pixel p at location (x, y). Consider its differences with all pixels that lie in the same row and column as pixel p at intervals of distance d. This provides an advantage over Scheme 1 by capturing structural properties in both the horizontal and vertical directions while restricting the dimensionality of the resulting feature vector. Using Scheme 2 results in a feature vector of length $M \times N \times (1 - \frac{M+N}{d})$.

We use Scheme 2 since we are able to accommodate the length of the resulting PoP feature vector.

The three classes of features are finally combined to form the resulting feature vector describing an image patch:

$$F = [F_{cmap} \quad F_{hog} \quad F_{pop}]. \qquad (2)$$

3 Partial Least Squares

The combination of the three feature classes results in an extremely high-dimensional feature space (approximately 70,000 dimensions). In contrast, the number of samples in our training data set is much smaller (about 200 in the positive and 1,500 in the negative class). Furthermore, our features are extracted from neighboring pixels within a detection window, which tremendously increases the correlation between them, rendering traditional Ordinary

Least Squares (OLS) regression estimates unreliable. This phenomenon is also known as the multicollinearity of the feature set. The nature of our proposed feature set makes an ideal setting for a statistical technique known as PLS regression [30].

The PLS method was first developed by Herman Wold in the 1960s and 1970s to address problems in econometric path-modeling [29], and was subsequently adapted in the 1980s to problems in chemometric and spectrometric modeling. In the late 1980s and 1990s, PLS attracted the attention of statisticians [10], [14] due to its ability to deal with a small number of examples and a large number of variables.

We present a brief introduction to PLS. For a more detailed analysis, see [3]. Consider a set of p predictor variables $X_1, X_2, \dots X_p$, which are used to predict q response variables $Y_1, Y_2, \dots Y_q$. Let n equal the number of observation pairs denoted as (x_i, y_i), where $\{i = 1, 2, \dots, n\}$. The data samples are assumed to be mean-centered. They are concatenated to form the matrices $X_{(n \times p)}$ and $Y_{(n \times q)}$. When $n < p$, classical regression tools cannot be applied since the covariance matrix $X^T X_{(p \times p)}$ is singular.

PLS regression is based on the following latent component decomposition:

$$X = TP^T + E, \tag{3}$$

$$Y = UQ^T + F, \tag{4}$$

where T and U give the latent components (known as the *scores* matrices), P and Q provide the coefficients (known as the *loadings* matrices), and E and F are the error matrices. Note that a decomposition similar to (3) is obtained by Principal Components Analysis.

The latent components given by T are obtained by a linear transformation of X as follows:

$$T_{n \times d} = X_{n \times p} W_{p \times d}, \tag{5}$$

where d is the dimensionality of the latent space. The latent components T are used for prediction in place of the original data vectors X. There are many variants of the basic PLS algorithm. They can be broadly classified based on their ability to deal with univariate response variables versus multivariate response variables. Multivariate response PLS has two popular implementations. The first variant leads to the NIPALS algorithm, whereas the second variant leads to the SIMPLS algorithm. These two methods differ in the matrix deflation process within the PLS algorithm. In our analysis, we have used the NIPALS algorithm. The NIPALS algorithm is essentially one of many methods that exist for finding the eigenvectors of a matrix. It was originally developed for Principal Components Analysis, but was subsequently used to iteratively extract factors for Partial Least Squares. Algorithm 1 provides a brief outline of the NIPALS algorithm. For more details, we refer the reader to [11].

Algorithm 1. NIPALS Algorithm

```
1: for $i = 1$ to $d$ do
2:     Matrix Projection
       $W_i = X^T Y / \|X^T Y\|$
       $T_i = X W_i / \|X W_i\|$
3:     Matrix Deflation
       $X = X - T_i T_i^T X$
       $Y = Y - T_i T_i^T Y$
4: end for
```

W_i and T_i represent the ith columns of the matrices W and T, respectively. The regression model is given by

$$Y = XB + F, \tag{6}$$

where $B_{p \times q}$ is the matrix of regression coefficients. Algebraic manipulations yield

$$B = WQ^T = W(T^T T)^{-1} T^T Y. \tag{7}$$

A new observation x_{new} thus yields a response value given by

$$y_{new} = \frac{1}{n} \sum_{i=1}^{n} y_i + B^T \left(x_{new} - \frac{1}{n} \sum_{i=1}^{n} x_i \right). \tag{8}$$

The data we are interested in fall into two classes—vehicles and background. We use the PLS regression algorithm as a *class-aware* dimensionality reduction tool by setting the class label of a sample in Y to 1 or -1. Thus, for our purpose, $q = 1$. Note that the matrix of regression coefficients B is now a single vector $(B_{p \times 1})$, with a single coefficient for every feature. In practice, we do not project a new observation onto B. Instead, we project it onto the first k columns of matrix W, and then apply a classifier on that subspace. This method allows us to apply any classifier within this subspace (linear or nonlinear) and has been shown to provide improved performance. The number of PLS factors k is obtained using cross validation.

Dimensionality reduction techniques can be broadly classified in two ways: linear versus nonlinear methods and supervised versus unsupervised methods. Nonlinear methods are generally computationally intensive and are not very suitable for extremely high-dimensional feature spaces such as ours. For the purposes of classification, supervised methods hold an advantage over unsupervised methods due to their use of the class information within the training data set.

Principal Component Analysis (PCA) is a classical linear unsupervised method. While PCA creates orthogonal latent vectors by maximizing the covariance between the data vectors x_i, PLS (a supervised approach) also considers the class labels. Fig. 5 demonstrates the advantage of PLS over PCA. A subset of the training data set (80 percent of the data points with all 69,552 variables) is provided to both PCA and PLS, and then projected onto the first two factors given by each dimensionality reduction method. The first row shows the training points projected onto the subspaces. The second row shows the test points (the remaining 20 percent of the data) projected onto the subspaces. The first two factors given by PLS are clearly more discriminating than PCA.

Fisher Discriminant Analysis (FDA) is in this way similar to PLS. It is a linear supervised method. However, FDA suffers from the *small sample size* problem. When the number of features exceeds the number of samples $(n < p)$, as in our case, the covariance estimates are not full rank, which are required to obtain the projection vectors. A number of

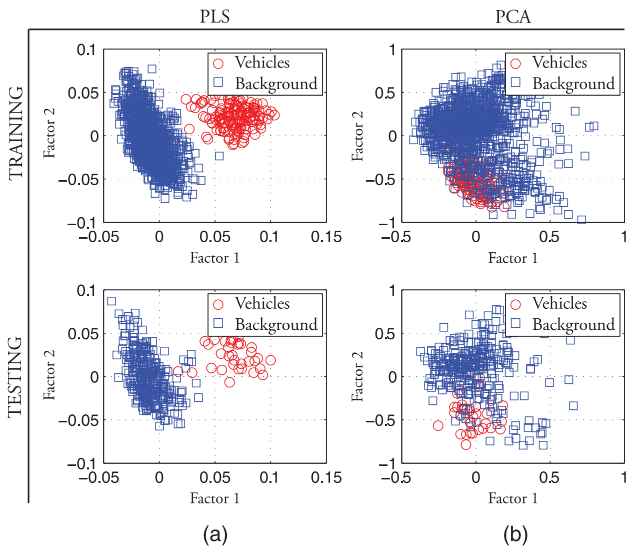

Fig. 5. Projection of data points from a 69,552D feature space onto a 2D subspace. In this illustration, 80 percent of the training data set are used to obtain the subspaces and the remaining 20 percent of the data from each class are used as the testing samples. The left column shows the subspace extracted using PLS. The right column shows the subspace extracted using PCA. Clearly, PLS extracts a subspace that is more discriminating than PCA.

extensions to LDA have been proposed to deal with this problem. Belhumeur et al. [1] first projected points onto a lower dimensional subspace using PCA (which does not suffer from the small sample size problem), and then applied FDA on the reduced subspace. Chen et al. [4] used a modified version of Fisher's criterion and proposed an efficient and stable algorithm to calculate the discriminant subspace. However, FDA has a further limitation, in that it retains only $l-1$ meaningful latent vectors, where l is the number of classes being considered. For our two-class problem, a 1D subspace might not be sufficiently discriminative.

3.1 Visualization of PLS Factors

It is useful to be able to visualize data points in the reduced PLS latent space. We propose a technique to visualize the separation between the data points in the two classes as the dimensionality of the PLS latent space increases.

Fig. 6 shows the visualization of data in a 5D PLS factor space. The top left plot shows the data plotted in the space spanned by the first two PLS factors. One can observe that the two classes are not completely separated in this 2D space. In order to visualize the effect of the third PLS factor, the data points are plotted in the space spanned by the third factor and the composite PLS dimension obtained from the first two PLS factors. The composite factor is essentially the vector of regression coefficients (B) given in (7). The data projected on B represent the output of PLS regression obtained using a 2D latent space. The top right plot shows the data projected on B as against the third PLS factor. In this way, data projected on factor i are plotted against the data projected on the composite factor obtained from the PLS factors 1 to $i-1$.

One can observe the increase in the separation between the two classes as the number of PLS factors is increased. This visualization is a useful tool to observe the structure of the data in the two classes in a high-dimensional PLS factor space.

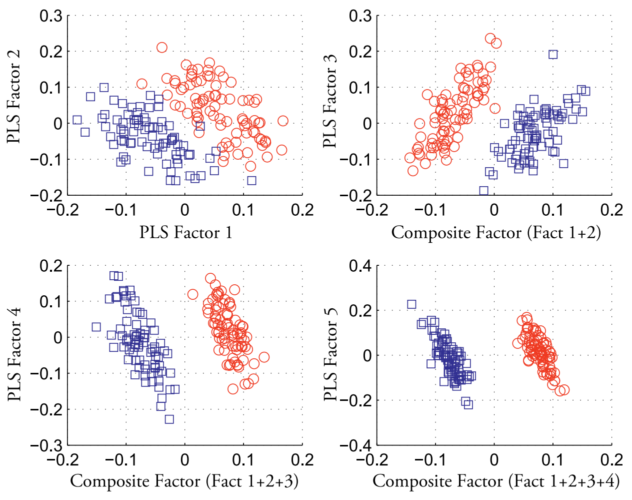

Fig. 6. Visualization of multiple PLS factors. The effect of the nth PLS factor can be visualized by plotting it against the composite factor obtained by combining the first $n-1$ PLS factors.

Note that it cannot be used to determine the optimum number of PLS factors for a given problem, which is obtained by a cross-validation procedure on the training data set.

4 Feature Selection

The large number of features greatly increases the computational cost of the vehicle detector. Calculating the features requires a substantial amount of time. At the same time, projecting a data point from an p-dimensional space down to a d-dimensional PLS factor space requires $p \times d$ multiplications and $(p-1) \times d$ additions. A typical image, which must be scanned at multiple orientations and scales, has hundreds of thousands of windows to be evaluated. Clearly, this would be a very time-consuming process. This can be greatly speeded up, without a significant loss in the performance, by using an effective feature selection process.

Furthermore, feature selection can often improve the performance of a PLS-based classification system. In our set of thousands of features, one can expect many of them to be very noisy and redundant. Feature selection can help discard a large fraction of such variables.

We use two methods to perform feature selection: Ordered Predictors Selection and a multistage Multiresolution Analysis.

4.1 Ordered Predictors Selection (OPS)

Variable Importance on Projection (VIP) is a widely used technique for PLS-based feature selection. VIP provides a score for every variable that ranks them according to their predictive power. A cross-validation scheme can then be used to select the number of variables required to obtain a desired level of classification accuracy. In general, any informative vector which provides a measure of the predictive power of the variables can be used for feature selection.

Teofilo et al. [27] used several informative vectors and their combinations to perform feature selection for regression problems. Their method is computationally efficient as compared to other variable selection methods, such as genetic algorithms, and can be completely automated. They used several different data sets in their analysis, some having a multiple number of dependent variables. Their analysis showed a rather surprising result. The number of PLS factors that were obtained by a cross-validation procedure for the purpose of feature selection was often higher than the number of PLS factors that were optimal for the purpose of regression.

We perform a similar feature selection analysis, enabling us to reject a large fraction of noisy features. We only deal with a single dependent variable, which is set to the class label $(+1/-1)$. The following informative vectors are used in our analysis.

1. VIP—The VIP score for the jth variable is a measure based on the weighted PLS coefficients. The higher the score, the more importance a variable presents. The average VIP score over all variables equals 1. The *thumb rule* used to select variables according to their VIP score is to retain only those variables whose VIP score is greater than 1:

$$\mathrm{VIP}_j = \sqrt{p \sum_{k=1}^{d} B_k^2 W_{jk}^2 \Big/ \sum_{k=1}^{d} B_k^2}. \tag{9}$$

2. Regression Coefficients (B)—The regression coefficients B defined in (7) represent the expected change in the response, per unit change in the variable. The absolute value of the regression coefficients is thus used as informative scores.
3. Correlation (CORR)—The correlation informative vector contains the Pearson correlation coefficients between every predictor variable X_i and the response variable Y. A high correlation indicates that the predictor variable is very informative about the PLS classification model:

$$\mathrm{R} = \frac{X^t Y}{n-1}. \tag{10}$$

Since the Pearson correlation coefficient lies between -1 and $+1$, with 0 indicating an absence of any correlation, the absolute value of the correlation coefficients is used.
4. Covariance Procedures Vector (CVP)—The CVP score was proposed by Reinikainen and Hoskuldsson [24] as a measure of variable importance. The ranking of variables is based on the covariance of the dependent and independent variables which is given by

$$\mathrm{CVP} = diag(X^t Y Y X). \tag{11}$$

Informative Vectors CORR and CVP do not depend on the dimensionality of the PLS factor space. However, VIP and BETA vary with the dimensionality of the factor space. Motivated by the OPS results obtained in [27], a nested cross-validation procedure is used to determine the optimum dimensionality of the PLS factor space for classification and the optimum dimensionality of the PLS factor space to calculate the informative vector.

Algorithm 2. OPS Cross Validation Procedure

```
1: for $\theta = 1$ to $N_{select}$ do
2:     Build PLS model using $\theta$ factors
3:     Calculate informative vector and sort variables by
       their informative score
4:     Choose top $K$ variables based on feature selection
       criteria
5:     for $\phi = 1$ to $N_{model}$ do
6:         Build PLS model using $\phi$ factors and top
           $K$ variables
7:         Calculate classification accuracy on the validation
           set
8:     end for
9: end for
```

4.2 Multistage Multiresolution Analysis

The nature of the three feature classes introduces a fair amount of redundancy in the feature set. For instance, the color probability maps capture color statistics of every pixel within an image patch. Clearly, neighboring pixels are highly correlated and introduce a lot of redundancy. Downsampling an image can help remove this redundancy somewhat at the cost of performance.

We build two PLS models using features computed from training images at different resolutions. The first model is built from training images reduced to a width and height that equal $1/2$ the original image dimensions, vastly reducing the dimensionality of the feature vector. The second model is computed from the original training images. The features that contribute toward each PLS model undergo the OPS feature selection strategy that was outlined in Section 4.1. Thus, we obtain two trimmed PLS models. In principle, one may add more downsampling stages to obtain a further speedup, possibly at the cost of accuracy.

Given a testing image, each and every image patch is classified using the first and fastest PLS model. A subset of these is then sent to the second stage which contains the second PLS model (lowest speed, highest performance).

5 EXPERIMENTS

5.1 Google Earth San Francisco Data Set

We test the performance of our vehicle detector on satellite images of the city of San Francisco taken from Google Earth. This data set consists of 40 satellite images at a resolution of $979 \times 1{,}348$ pixels and a color depth of 24 bits per pixel (RGB). Fig. 7 shows one such image. Each image looks down on an urban scene with multiple cars present. The total number of vehicles in the data set is 650. The average size of a vehicle is 48×16 pixels.[1]

The first five images in the data set are used for training and the performance of our vehicle detector is tested on the remaining 35 images. One hundred eighty-four positive image patches (vehicles) are extracted from the training images and aligned vertically. Similarly, 1,500 negative image patches are randomly chosen from the training images. Each image patch has a size of 81×41 pixels. Fig. 7 shows a few training image patches from the data set. The test images are scanned using a sliding window approach.

1. The highest resolution for current commercial satellite imagery is 0.5 meters, whereas the images used in the San Francisco data set correspond to a resolution of 0.1 meters. We have used images captured using the software: Google Earth. Images provided by Google Earth beyond this resolution are obtained by digitally enlarging the images.

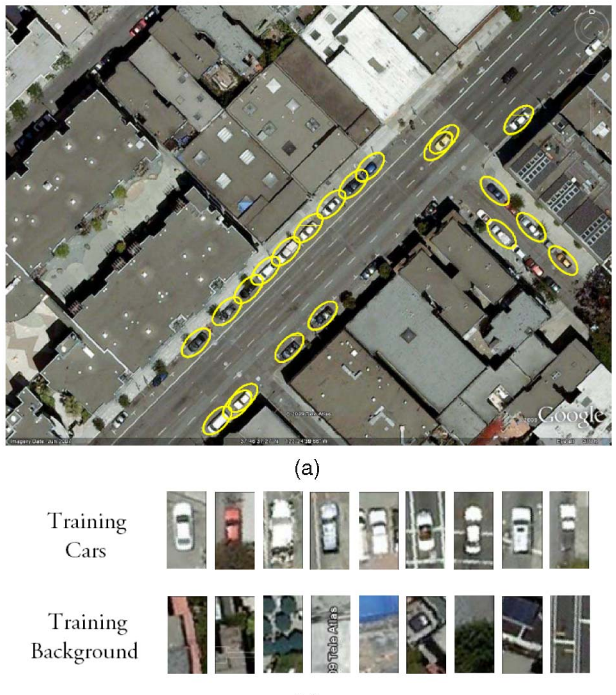

Fig. 7. Example images from the Google Earth San Francisco data set. (a) A test image from the data set, looking down on an urban scene, with overlaid detections using our vehicle detector. (b) A few training images patches from both classes. © 2009 Google.

The size of the window is fixed to 81×41 pixels since the training and test images have been captured at the same resolution and all vehicles are observed at a single scale. More generally, however, one may need to scan the image at multiple scales. The horizontal and vertical step sizes are set to 5 pixels.

5.2 Feature Extraction and Evaluation

Each image patch has a size of 81×41 pixels. Six color clusters are used to build our color probability maps. This results in a feature vector of length 19,926. The HOG feature is calculated for 661 blocks ranging from a size of 12×12 pixels to a size of 80×40 pixels. Each block yields a 36D feature vector resulting in a total length of 23,796. The PoP feature is calculated using images reduced to size 41×21, which gives a feature of length 25,830. Thus, the length of the resulting feature vector obtained by combining the three feature classes is 69,552.

All the parameter selection in our system is performed by using three iterations of a fivefold cross-validation scheme. The analysis presented in Figs. 8, 9, 10, 11, 12, 13, 14, and 15 was carried out using this cross-validation procedure on the Google Earth San Francisco training data set.

Fig. 8 shows the mean classification error obtained using each of the three feature classes separately as well as their combination. As the number of PLS factors is increased, the classification error reduces. Beyond a certain value, however, addition of PLS factors does not yield an improved

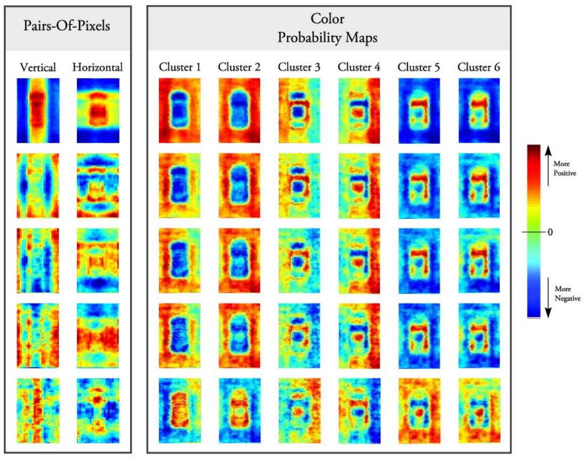

Fig. 9. The first five PLS latent vectors. The components of the latent vectors corresponding to the Color Probability Maps and Pairs of Pixels feature classes are displayed. The images shown for the color probability maps directly correspond to the latent vectors. The images displayed for the PoP feature correspond to accumulator matrices. *(best viewed in color)*.

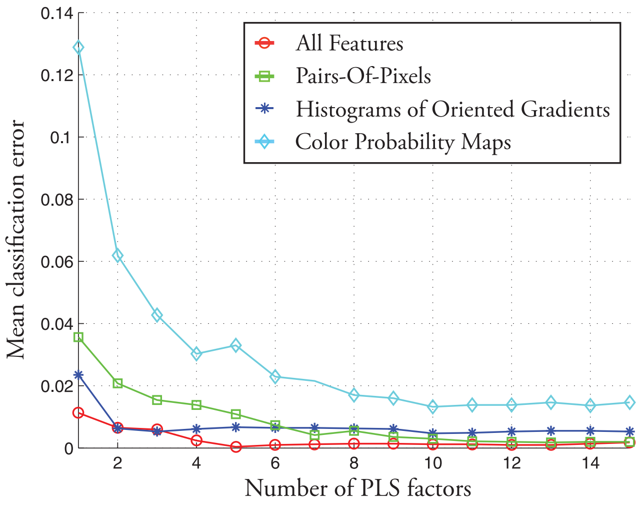

Fig. 8. Mean classification error obtained after three iterations of five-fold cross validation on the Google Earth San Francisco training data set. The error plot is shown for each class of features individually, as well as their combination.

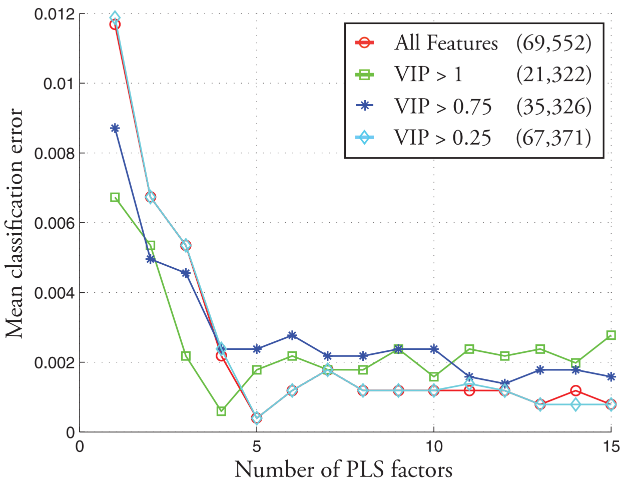

Fig. 10. Basic feature selection using VIP. The error plot is shown for the original set of features and for a subset of features using the VIP criterion. VIP > 1 is the *thumb rule* usually applied in PLS analysis.

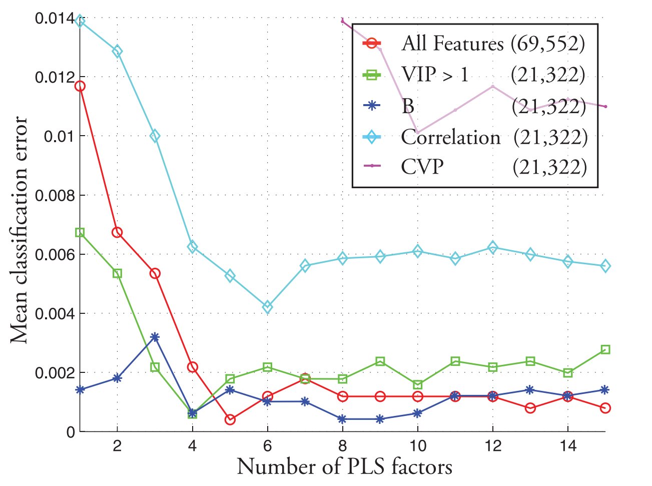

Fig. 11. Basic feature selection using all four informative vectors. The regression vector B, when used as an informative vector, outperforms the typically used VIP informative vector. The CVP vector performs very poorly on our data set.

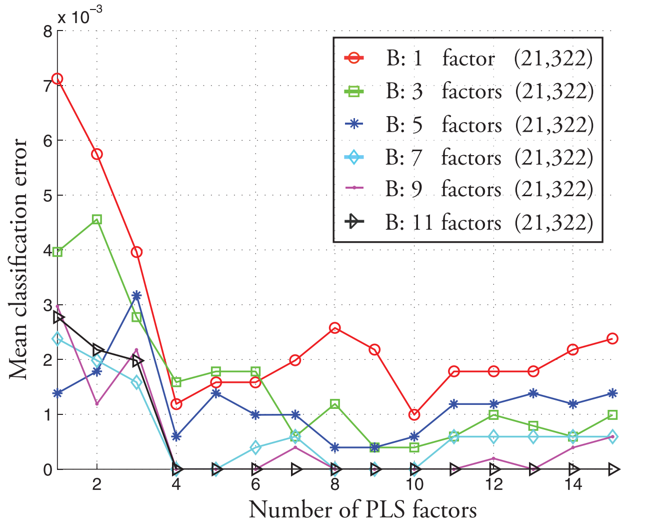

Fig. 12. Feature selection using the B informative vector. The number of PLS factors used to calculate B (θ) is varied. The error is seen to reduce as θ is increased and it saturates beyond 11.

performance. This saturation point is generally regarded as the optimum number of PLS factors. In some cases, performance decreases as more PLS factors are introduced. The PoP feature class outperforms the other two feature classes, but the best performance is obtained when all three

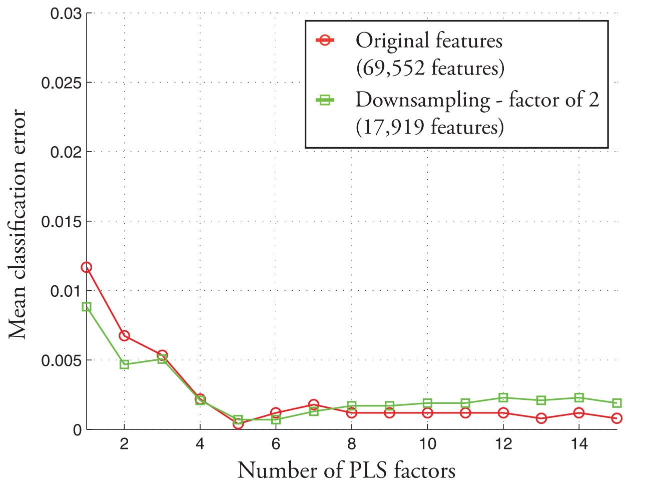

Fig. 14. Mean classification error obtained using downsampling. No feature selection is used. The performance decreases very slightly as images are reduced to a smaller size.

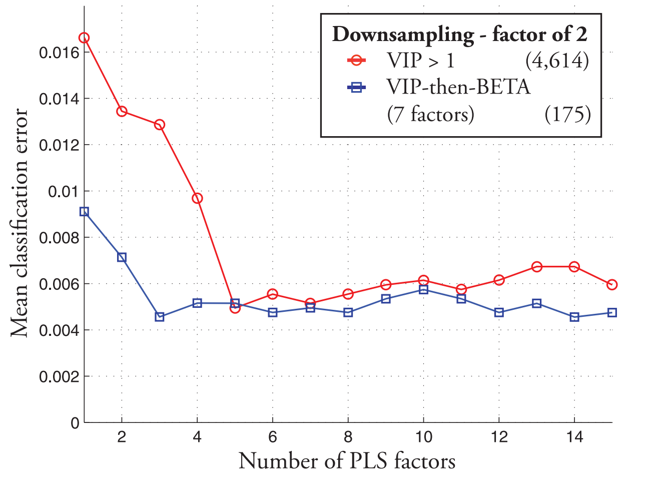

Fig. 15. Mean misclassification error at downsampling factor of 2. The error rates drop after the feature selection process. The informative vector combination VIP-then-B outperforms VIP > 1.

feature classes are combined. The number of factors chosen, when the entire set of features is used, is 5.

Fig. 9 shows the components of the first five PLS latent vectors (Ws) that correspond to the color probability maps and the PoP features. The latent vectors corresponding to the color probability maps are directly displayed, as shown in the figure, since each coefficient corresponds to a single pixel location in the image patch. But each coefficient of the

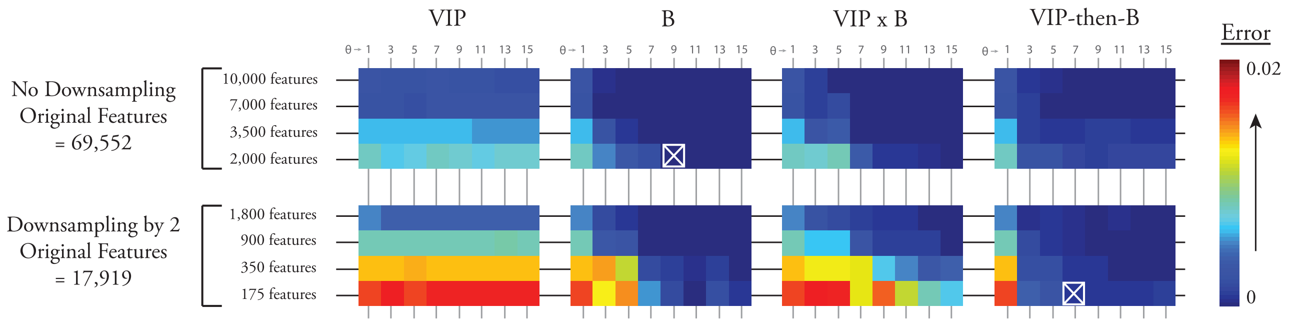

Fig. 13. Results of the OPS feature selection approach on the original training set and the downsampled set. The errors represent the mean misclassification error after cross validation. (Refer to the text for more details.)

latent vector from the PoP features corresponds to two pixel locations in the image (a pair of pixels), and thus cannot be displayed directly. Instead, the images displayed in the figure for the PoP feature class correspond to accumulator matrices. We initialize the accumulator matrix as a matrix of zeros and increment a location with the corresponding coefficient in the latent vector, when that location forms one half of the corresponding pixel pair. Furthermore, the images corresponding to the PoP feature class are split into the vertical and horizontal components. Each coefficient in the HOG feature vector corresponds to a bin of a histogram and captures information about a neighborhood of pixels. These neighborhoods also have varying sizes (multiple block sizes), and thus cannot be easily visualized as the other two feature classes.

A very positive (dark red) or very negative (dark blue) color indicates a high degree of importance afforded to that pixel location. It is noticeable that the feature classes complement each other well, by extracting information from different parts of the image patch. The first (and most important) factor for the color probability maps shows that information extracted from the neighborhood of the object is given a high weight. This indicates that these features are able to capture the immediate context within which vehicles are typically observed. The PoP feature is seen to primarily focus on pixels that lie on top of the vehicle, in order to capture its regular structure.

5.3 Feature Selection: OPS

Fig. 10 shows the result of the commonly used feature selection criterion using VIP. The VIP informative vector has been calculated using five PLS factors (which was the optimal number when using the entire set of features—refer to Fig. 8). Using the $\text{VIP} > 1$ thumb rule reduces the number of features to 21,322, which represents about 30 percent of the total feature set. The performance does not drop significantly; instead a comparable performance is obtained using a smaller number of PLS factors. Results using varying values of the VIP cut-off score are shown.

Fig. 11 shows the results of feature selection using the four informative vectors. The VIP and B informative vectors have been calculated using five PLS factors. The Correlation and Covariance Procedures Vector are calculated independent of the number of PLS factors. As a fair comparison, we use the same number of features with each informative vector. The number 21,322 is the number of features retained using the $\text{VIP} > 1$ rule. The best performing informative vector is the regression vector B.

Fig. 12 shows the results of feature selection using the B informative vector when the number of PLS factors (θ) used to compute this vector is varied. As was observed in [27], the optimum number of PLS factors used to calculate the informative vector (θ) is not the same as the optimum number of PLS factors used to build the model (ϕ), when all features are used (ϕ was determined to be 5 in our case). The performance is shown to increase with θ and it saturates beyond 11. The number of features selected was, once again, set to the number obtained by the $\text{VIP} > 1$ rule. The mean error after cross validation is seen to go down to 0 when B is calculated from 11 PLS factors and 21,322 features (≈ 30 percent of the total set) are retained.

Fig. 13 shows the classification errors obtained using different informative vectors and their combinations. Eight matrices are displayed in the figure in a 2×4 grid. Each column in the grid of matrices corresponds to the results obtained for a particular informative vector. Each row in the grid of matrices corresponds to the results obtained for particular set of input features. We first consider only the first row in this grid of matrices, where all 69,552 features are provided as input to the feature selection module. We discuss the second row in Section 5.4. Each matrix in the 2×4 grid is color-coded to represent the classification errors obtained by the cross-validation procedure. Each matrix column represents a different value of θ. In general, as θ increases, the classification error reduces, and then, saturates. Each matrix row represents a different number of features retained after the feature selection process. As this number reduces, the error increases. Since the Correlation and CVP informative vectors do not perform very well, they are not displayed in this image.

B is the most consistent and best performing informative vector. It also performs very well when just 2,000 features of the total 69,552 features are retained. This corresponds to less than 3 percent of the total number of features. The informative vectors are usually combined by simply multiplying their corresponding bins. The combination of VIP and B outperforms other combinations, and is the only one displayed in this figure. We also introduce a new scheme to combine the B and VIP informative vectors called *VIP-then-B*. First, the $\text{VIP} > 1$ rule is employed to select a set of features. Then, a new PLS model is built using these chosen features and the informative vector B is calculated. This smaller set of features is then ranked using B and a subset of these is finally selected. *VIP-then-B* is seen to perform very well, especially when small sets of features are chosen. We finally choose the following feature selection scheme: B with $\theta = 9$ and 2,000 features selected (over all three feature classes). Cross validation determined the number of PLS factors in the model (ϕ) to be 6. The cell corresponding to this parameter choice is marked with the white cross.

5.4 Feature Selection: Downsampling

The original image patches have a size of 81×41 pixels. The total number of features computed for these original image patches equals 69,552. These image patches are downsampled by a factor of 2 (41×21 pixels). This reduces the total number of features to 17,919 (5,166 color, 9,288 HOG, and 3,465 PoP features).

Fig. 14 shows the result of downsampling without any feature selection. As is seen, performance does not degrade too much for a downsampling factor of 2. This reduced set of features is used in the first stage of our two-stage PLS model. This enables us to efficiently reject a large number of background image patches and only pass on a small set of candidate vehicles to the next stage.

For each level of downsampling, a thorough OPS-PLS analysis is carried out using cross validation on the training data set. These results are displayed in the second row of the 2×4 grid of matrices displayed in Fig. 13. This determines the informative vector chosen for feature selection and the number of parameters associated with it.

TABLE 1
The Two-Stage Vehicle Detector

	STAGE 1	STAGE 2
Image size	41×21 pixels	81×41 pixels
Original features	17,919	69,552
Feature selection	VIP-then-B ($\theta = 7, \phi = 7$)	B ($\theta = 9, \phi = 6$)
Retained features	175	2,000
Percentage Windows Processed	100%	0.48%

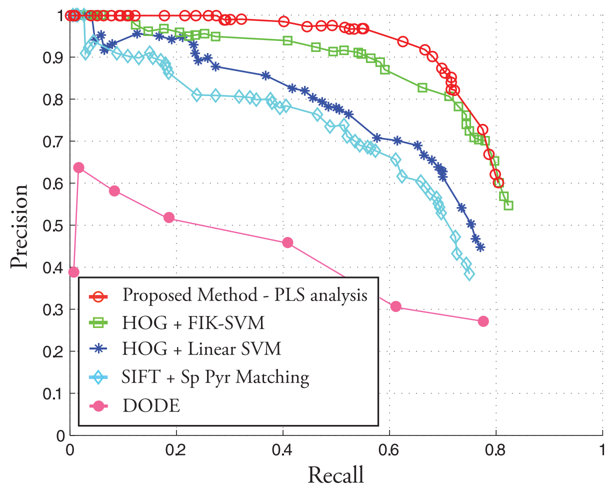

Fig. 16. Performance of the five vehicle detectors on the Google Earth San Francisco data set. Our vehicle detector outperforms the other ones.

For a downsampling factor of 2, the VIP-then-B informative vector combination provides the best performance. Remarkably, the feature selection strategy retains only 175 features in the fast first stage (almost 0.25 percent of the original number of features 69,552). This enables a fast rejection strategy and improves the performance of the system. Fig. 15 shows the performance at a downsampling factor of 2 when the popular VIP > 1 feature selection strategy is used, as well as when the best feature selection criteria are used. Thus, we obtain a two-stage PLS model, whose parameters are given in Table 1.

5.5 Performance: Google Earth San Francisco Data Set

We test the performance of our system on the test images of the Google Earth San Francisco data set. All test images are fully ground truthed to show the presence of vehicles along with their orientation. Vehicle detections that overlap the ground truth locations by an area equal to 33 percent of the size of the bounding box are considered true detections. As per the evaluation criteria commonly used in the PASCAL VOC challenge [8], if multiple detections overlap with a ground truthed location, only one of them is considered a true positive detection. The remaining detections are considered to be false alarms. Since the vehicles in the entire data set are roughly the same size, we only scan the images at a single scale. If the size of the vehicles was unknown, the images would have to be scanned at multiple scales. Since the orientation of the vehicles is unknown, the image must be scanned at multiple rotations. In order to decrease the number of image patches that must be scanned, we employ a coarse to fine rotation strategy. First, the image is rotated in increments of 30 degrees and scanned completely. Candidate windows are selected and are then finely rotated in increments of 5 degrees in a neighborhood of 25 degrees from the original detection.[2] In an urban setting, roads typically form a rectangular grid. Determining this grid orientation can help to initialize the scan angles and further speed up the system.

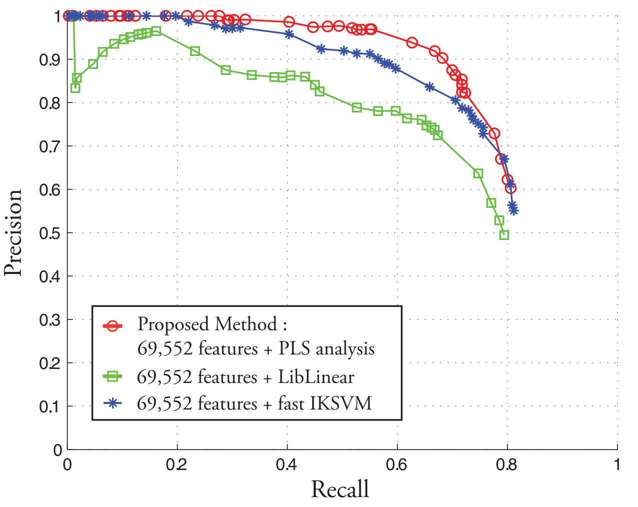

Fig. 17. Performance of the proposed vehicle detector compared to detectors composed of the proposed features and SVMs as classifiers.

TABLE 2
Comparison of Classification Speeds (Windows/Second) While Using 69,552 Features

	PLS method	LibLinear	Fast IKSVM
Detection speeds (win/sec)	8565	308	95

We compare our system to a number of other approaches. The first approach is a traditional object detection approach.[3] SIFT features are calculated on a dense grid of points in the training images. The SIFT descriptors from the training set undergo vector quantization to form visual words. An image patch can then be described using a histogram of the visual words present in the patch. In order to enforce some spatial constraints on the location of these features, we use Spatial Pyramidal Matching [16]. This involves repeatedly subdividing the image and computing histograms of SIFT features at increasingly finer resolutions. The distance measure used is histogram intersection and the classifier used is the Support Vector Machine.

The second approach we compare to is the vehicle detector proposed by Moon et al. [20]. They derive an optimal 1D step edge detector to be the DODE function. This is extended along the shape's boundary contour to obtain the shape detector.[4] This results in detecting shapes in the image that resemble parallelograms. Their vehicle detector does not require a training phase.

The third approach we compare to is the popularly used HOG-based approach used to detect objects, proposed by Dalal and Triggs [6]. HOG features are calculated for a large number of blocks within an image window and

2. The identical scanning strategy is used for the proposed vehicle detector and the approaches we compare to.
3. Code obtained from http://www.cs.unc.edu/lazebnik/.
4. Code obtained from the authors.

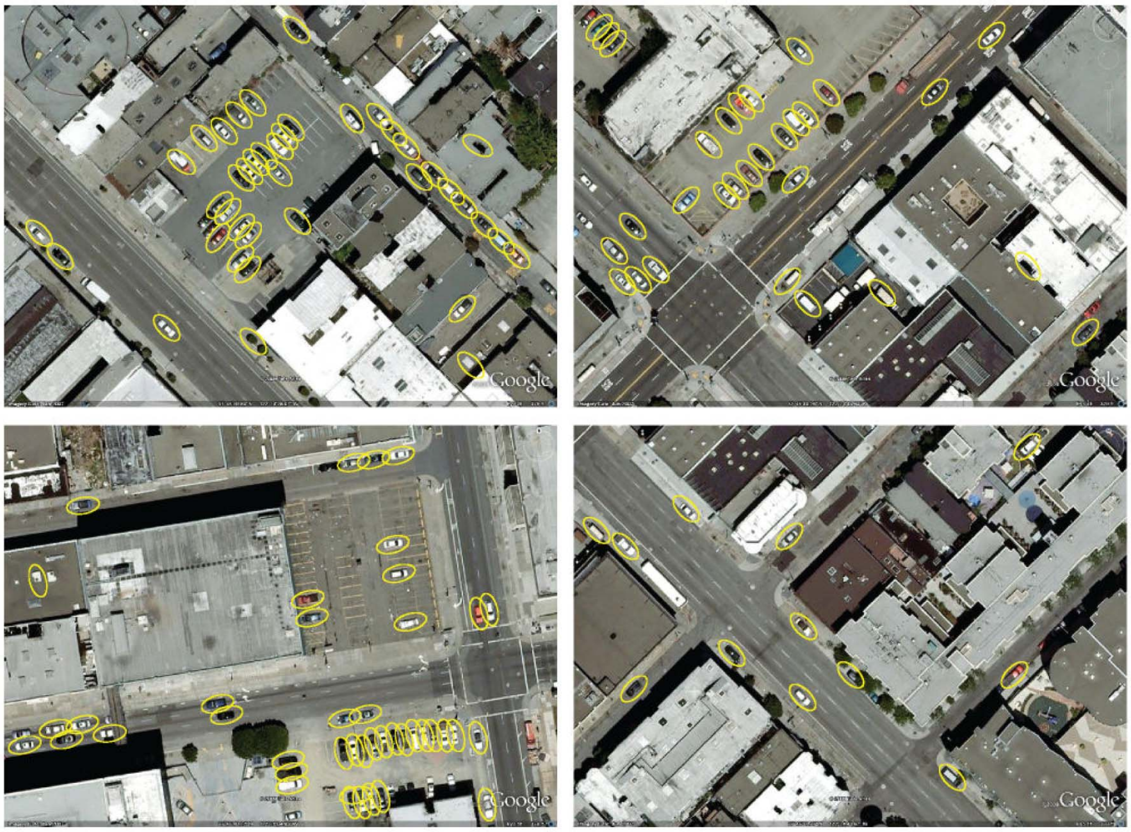

Fig. 18. Sample vehicle detection results from the Google Earth San Francisco data set using our proposed approach.

concatenated together to form the feature vector. Blocks of only a single size are used in this approach. The feature vectors thus obtained are used to train a linear Support Vector Machine (SVM). We used libLinear [9],[5] an efficient linear SVM, for this purpose. Dalal et al. note in their work that a kernel SVM can provide improved performance at the cost of a significant reduction in the efficiency of the system. This is because the standard approach to evaluate the kernel for a test vector involves a comparison with each of the support vectors.

Recently, Maji et al. [19] proposed a method to significantly improve the efficiency of kernel SVMs for a class of kernels such as the histogram intersection kernel and the chi squared kernel. Their approximate SVM has constant runtime and space requirements, independent of the number of support vectors, as opposed to the typical linear dependency. Furthermore, the loss in classification accuracy using this approximation is negligible. As a fourth comparison, we applied this improved kernel SVM[6] known as the approximate intersection kernel support vector machine (approx IKSVM) to the HOG features proposed by Dalal and Triggs. While [19] demonstrated that the IKSVM evaluates nearly as fast as a linear classifier, it did not address the problem of efficiently training such a classifier. Subsequently, Maji and Berg [18] proposed very efficient training algorithms for additive classifiers in a max-margin framework.

5. Package available at http://www.csie.ntu.edu.tw/cjlin/liblinear/.
6. Code obtained from http://www.cs.berkeley.edu/smaji/projects/fiksvm/.

Fig. 16 shows the Precision-Recall curves for the vehicle detectors. The data set we test on contains images in an urban setting. The presence of a large number of rectilinear structures in such images leads to false alarms with all approaches. Objects present on top of buildings, such as air conditioning units, are seen to cause errors. The DODE approach is able to correctly find many vehicles but also produces a very large number of false alarms on rectangular structures. The HOG features when applied to a linear SVM outperform the SIFT-based approach. The use of the

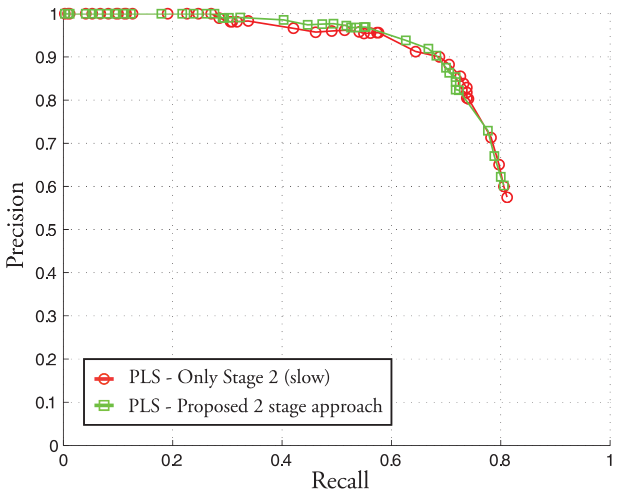

Fig. 19. Performance of the two-stage vehicle detector compared to the performance of the detector when only the high-resolution stage (Stage 2) is used.

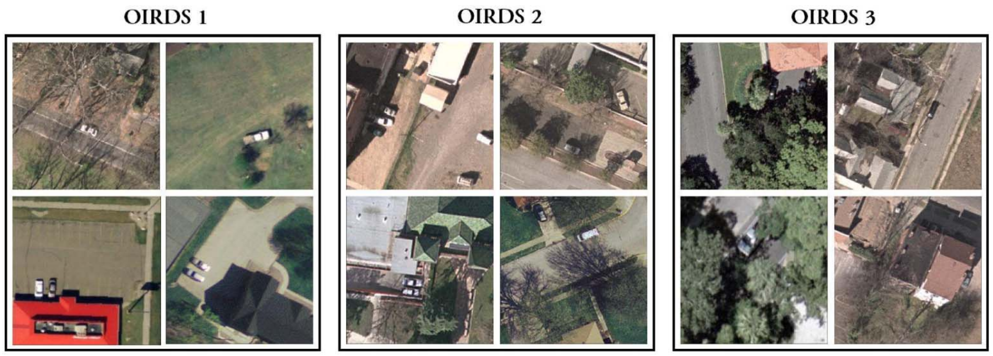

Fig. 20. Sample images taken from the OIRDS data set. We divide this data set into three parts based on the degree of difficulty.

histogram intersection kernel greatly improves the performance over the linear kernel. Our proposed solution outperforms all of the other approaches.

Fig. 17 compares our proposed approach to directly applying an SVM to our proposed set of 69,552 features. The PLS-based approach comfortably outperforms the approach using a linear SVM. We used the LibLinear package for this purpose. Using the approximate and fast IKSVM method improves results over using a linear kernel. However, applying the IKSVM to the original 69,552 features is quite time-consuming. Table 2 compares the speeds of the three classification approaches, as shown in Fig. 17. Note that the numbers provided in Table 2 account for the classification time only, and not the time required to calculate the features. Projection onto the subspace followed by classification by a quadratic classifier is very fast compared to the other two approaches. Applying the fast approximate intersection kernel is slower than the other two methods.

Overall, our two-stage vehicle detection system is able to process approximately 5,600 detection windows per second. Our system is implemented in MATLAB and all of our experiments are run on an Intel Xeon 2.8 GHz processor. While the machine we use has multiple processing cores, our program currently makes use of only a single core. The number of detection windows processed per second can be improved by taking advantage of multicore architectures. All three stages of our detection system—feature calculation, projection onto a subspace as well as classification—can be parallelized to yield a faster detection system.

5.6 Speedup Using Feature Selection

Fig. 19 compares the performance of the two-stage vehicle detector with a detector when only the high-resolution stage (Stage 2) is used. The Precision-Recall curves of the two detectors are quite comparable. This enables us to obtain the speedup given by the two-stage approach, without a significant loss of performance.

Table 1 shows the percentage of windows that are processed in each stage, over the entire Google Earth San Francisco data set. All windows are processed by the fast Stage 1, but only a very small number are passed on to the second stage. Note that the image patches that obtain a high detection probability in stage 2 undergo further processing by rotating them at finer angular intervals. This processing is done at the full resolution.

5.7 Overhead Imagery Research Data Set

We also test the performance of our system on the publicly available Overhead Imagery Research Data Set (OIRDS).[7] The OIRDS is a large collection of almost 900 overhead images, captured using aircraft-mounted cameras. The total number of vehicles annotated in the data set is around 1,800.

The OIRDS data set is a very challenging data set. The images in this set have varying levels of zoom and a wide degree of difficulty. The images have been captured primarily in suburban settings. The presence of a large number of trees in these images often causes vehicles to be partially occluded. We divide this data set into three parts (with roughly 300 images in each part). We label these parts OIRDS 1, OIRDS 2, and OIRDS 3.[8] OIRDS 1 contains images that have the best picture quality and vehicles that are clearly visible. These images are similar in quality to the images in the Google Earth San Francisco data set. OIRDS 2 contains images that are of a poorer quality and the vehicles are also harder to find. Some of these vehicles are partially occluded. OIRDS 3 contains images in which vehicles are very difficult to find. Many of these images have vehicles that were almost fully occluded. Fig. 20 shows sample images from these three sets.

We compare the performance of the five vehicle detectors on OIRDS 1 and OIRDS 2. No new training was carried out for this data set. The detectors trained on the Google Earth San Francisco training data set were directly used. Figs. 21 and 22 show the Precision-Recall curves for all three detectors. We outperform all detectors on OIRDS1 and obtain a comparable performance to the $\mathrm{HOG} + \mathrm{kernel}$ SVM approach on OIRDS 2. As expected, the performance of all the detectors drops for OIRDS 2.

Finally, Fig. 23 shows the performance of our vehicle detector on a large panoramic image ($5{,}007 \times 7{,}776$ pixels). This was obtained by stitching a large number of images obtained from Google Earth. The image overlooks the parking lot and adjacent areas of the San Francisco Giants stadium in San Francisco city. Our vehicle detector is able to accurately locate a large number of vehicles in this image.

5.8 Misclassifications

Figs. 18 and 23 show the presence of a few missed detections as well as false alarms. The presence of stark

7. Downloaded from http://sourceforge.net/apps/mediawiki/oirds.
8. More details available at: www.umiacs.umd.edu/ani/vehicleDetection.html.

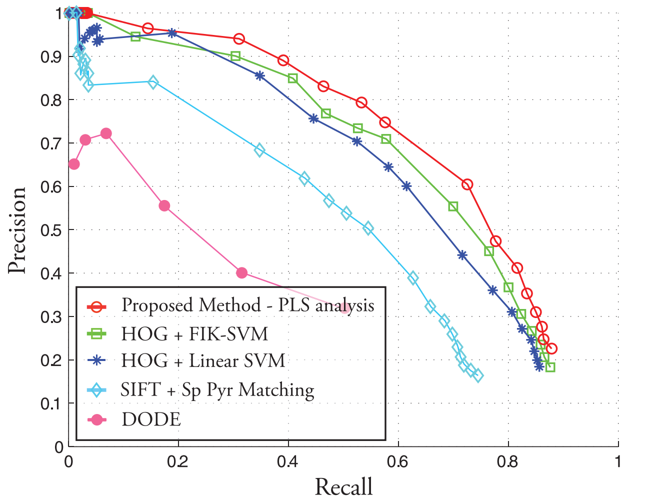

Fig. 21. Performance of the three vehicle detectors on the OIRDS 1 data set.

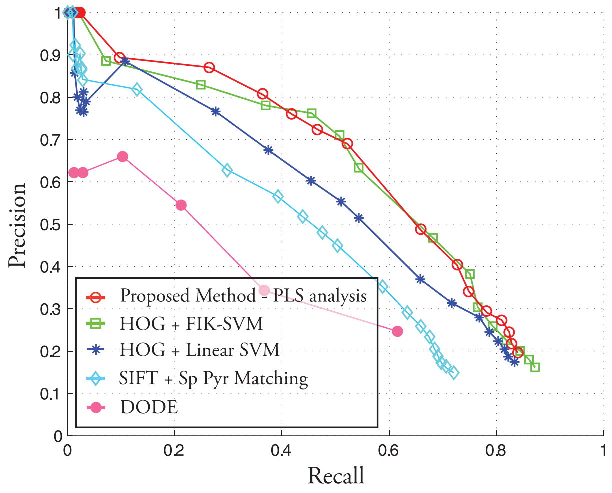

Fig. 22. Performance of the three vehicle detectors on the OIRDS 2 data set.

and long shadows often causes corruption of the measured feature vectors and lead to missed detections. Similarly, saturated pixels on the body of vehicles, caused by the reflection of the sun, also give rise to missed detections. Some dark cars are also missed by the vehicle detector due to the difficulty in capturing the structure present within the body of the vehicle (such as windows). Since the algorithm densely scans the image at multiple locations and orientations, we employ nonmaximal suppression to only retain detections that are dominant within a given radius. However, this also leads to the removal of some true detections. False alarms are typically caused by rectangular structures on top of buildings such as air conditioning units. Markings on the road such as the ones to denote parking spots may also lead to some false alarms.

Vehicles typically follow a small set of patterns when they are stationary as well as moving. For example, they are typically present on roads or parking lots; parked cars usually align to lie parallel to each other; cars on the road and in the same lane usually move in the same direction. Using such contextual cues (as demonstrated in [13]) could

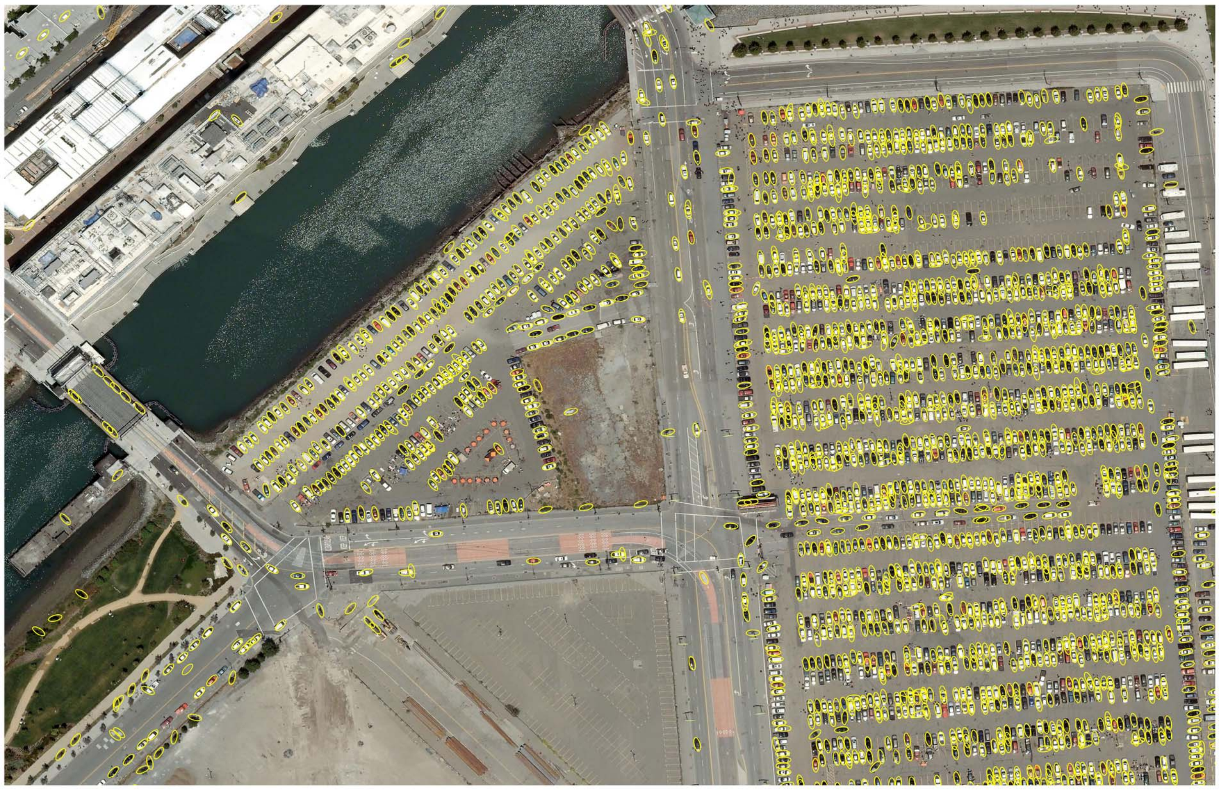

Fig. 23. Performance of our vehicle detector on a large panoramic image overlooking the parking lot of the San Francisco Giants stadium. © 2009 Google.

further reduce the number of false detections in a given image, as well as improve the orientation estimation of the detected vehicles.

6 Conclusion

We propose a vehicle detection system that incorporates a large set of rich features capturing color, gradient, and structural properties of vehicles and their surroundings. A Partial Least Squares analysis enables us to project points from a very high-dimensional feature space on to a low-dimensional subspace. We experiment with a number of informative vectors which quantify the discriminative power of individual features. This allows us to choose a small subset of features, while discarding many noisy features. The informative vectors which produced the best results were the set of regression coefficients (B), the variable importance on projection vector (VIP), and their combination VIP-then-B. We further speed up our system by employing a multistage, multiresolution strategy to quickly reject a large fraction of image patches using a cheaper set of features and only pass on a small set of candidate patches for further analysis. We show superior performance to previous approaches on two data sets: a data set of satellite images overlooking the city of San Francisco, and OIRDS, a publicly available data set of aerial images.

References

[1] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711-720, July 1997.

[2] A. Berg and J. Malik, "Geometric Blur for Template Matching," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, 2001.

[3] A.-L. Boulesteix and K. Strimmer, "Partial Least Squares: A Versatile Tool for the Analysis of High-Dimensional Genomic Data," *Briefings in Bioinformatics*, vol. 8, no. 1, pp. 32-44, 2007.

[4] L.-F. Chen, H.-Y.M. Liao, M.-T. Ko, J.-C. Lin, and G.-J. Yu, "A New Lda-Based Face Recognition System Which Can Solve the Small Sample Size Problem," *Pattern Recognition*, vol. 33, pp. 1713-1726, 2000.

[5] J.-Y. Choi and Y.-K. Yang, "Vehicle Detection from Aerial Images Using Local Shape Information," *Proc. Third Pacific Rim Symp. Advances in Image and Video Technology*, 2008.

[6] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, 2005.

[7] L. Eikvil, L. Aurdal, and H. Koren, "Classification-Based Vehicle Detection in High-Resolution Satellite Images," *ISPRS J. Photogrammetry and Remote Sensing*, vol. 64, no. 1, pp. 65-72, 2009.

[8] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results," http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html, 2008.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A Library for Large Linear Classification," *J. Machine Learning Research*, vol. 9, pp. 1871-1874, 2008.

[10] P.H. Garthwaite, "An Interpretation of Partial Least Squares," *J. Am. Statistical Assoc.*, vol. 89, no. 425, pp. 122-127, 1994.

[11] P. Geladi and B. Kowalski, "Partial Least-Squares Regression: A Tutorial," *Analytica Chimica Acta*, vol. 185, pp. 1-17, 1986.

[12] H. Grabner, T. Nguyen, B. Gruber, and H. Bischof, "On-Line Boosting-Based Car Detection from Aerial Images," *ISPRS J. Photogrammetry and Remote Sensing*, vol. 63, no. 3, pp. 382-396, 2008.

[13] G. Heitz and D. Koller, "Learning Spatial Context: Using Stuff to Find Things," *Proc. European Conf. Computer Vision*, 2008.

[14] I. Helland, "On the Structure of Partial Least Squares," *Comm. in Statistics., Simulation, and Computation*, vol. 17, pp. 581-607, 1988.

[15] S. Hinz, "Detection and Counting of Cars in Aerial Images," *Proc. Int'l Conf. Image Processing*, 2003.

[16] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Match for Recognizing Natural Scene Categories," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, 2006.

[17] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[18] S. Maji and A.C. Berg, "Max-Margin Additive Classifiers for Detection," *Proc. IEEE Int'l Conf. Computer Vision*, 2009.

[19] S. Maji, A.C. Berg, and J. Malik, "Classification Using Intersection Kernel Support Vector Machines Is Efficient," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[20] H. Moon, R. Chellappa, and A. Rosenfeld, "Optimal Edge-Based Shape Detection," *IEEE Trans. Image Processing*, vol. 11, no. 11, pp. 1209-1227, Nov. 2002.

[21] H. Moon, R. Chellappa, and A. Rosenfeld, "Performance Analysis of a Simple Vehicle Detection Algorithm," *Image and Vision Computing*, vol. 20, no. 1, pp. 1-13, 2002.

[22] V.I. Morariu, B.V. Srinivasan, V.C. Raykar, R. Duraiswami, and L.S. Davis, "Automatic Online Tuning for Fast Gaussian Summation," *Proc. Advances in Neural Information Processing Systems*, 2008.

[23] V.C. Raykar and R. Duraiswami, "Fast Optimal Bandwidth Selection for Kernel Density Estimation," *Proc. SIAM Int'l Conf. Data Mining*, 2006.

[24] S. Reinikainen and A. Hoskuldsson, "Covproc Method: Strategy in Modeling Dynamic Systems," *J. Chemometrics*, vol. 17, pp. 130-139, 2003.

[25] C. Schlosser, J. Reitberger, and S. Hinz, "Automatic Car Detection in High Resolution Urban Scenes Based on an Adaptive 3d Model," *Proc. Second GRSS/ISPRS Joint Workshop Remote Sensing and Data Fusion over Urban Areas*, 2003.

[26] F. Tanner, B. Colder, C. Pullen, D. Heagy, M. Eppolito, V. Carlan, C. Oertel, and P. Sallee, "Overhead Imagery Research Data Set an Annotated Data Library and Tools to Aid in the Development of Computer Vision Algorithms," *Proc. IEEE Applied Imagery Pattern Recognition Workshop '09*, 2009.

[27] R. Teofilo, J. Martins, and M. Ferreira, "Sorting Variables by Using Informative Vectors as a Strategy for Feature Selection in Multivariate Regression," *J. Chemometrics*, vol. 23, pp. 32-48, 2009.

[28] P. Viola and M. Jones, "Robust Real-Time Object Detection," *Int'l J. Computer Vision*, 2002.

[29] H. Wold, "Estimation of Principal Components and Related Models by Iterative Least Squares," *Multivariate Analysis*, Academic Press, 1966.

[30] H. Wold, "Partial Least Squares," *Encyclopedia of Statistical Sciences*, S. Kotz and N. Johnson, eds., vol. 6, pp. 581-591, Wiley, 1985.

[31] Z. Yue, D. Guarino, and R. Chellappa, "Moving Object Verification in Airborne Video Sequences," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 19, no. 1, pp. 77-89, Jan. 2009.

[32] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study," *Int'l J. Computer Vision*, vol. 73, no. 2, pp. 213-238, 2007.

[33] T. Zhao and R. Nevatia, "Car Detection in Low Resolution Aerial Images," *Image and Vision Computing*, vol. 21, no. 8, pp. 693-703, 2003.

[34] H. Zheng, L. Pan, and L. Li, "A Morphological Neural Network Approach for Vehicle Detection from High Resolution Satellite Imagery," *Proc. Int'l Conf. Neural Information Processing*, 2006.

[35] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, 2006.

Aniruddha Kembhavi received the bachelor's degree in electronics and telecommunications engineering from the Government College of Engineering, Pune, India, in 2004. He is currently working toward the PhD degree at the Computer Vision Laboratory, University of Maryland, College Park. His current research focuses on the problem of variable and feature selection for object classification. His research interests include human detection, object classification, and scene understanding.

David Harwood received the graduate degrees from the University of Texas and the Massachusetts Institute of Technology. His research, with many publications, is in the fields of computer image and video analysis and AI systems for computer vision. He is a longtime member of the research staff of the Computer Vision Laboratory of the Institute for Advanced Computer Studies at the University of Maryland College Park. He is a member of the IEEE.

Larry S. Davis received the BA degree from Colgate University in 1970, and the MS and PhD degrees in computer science from the University of Maryland in 1974 and 1976, respectively. From 1977 to 1981, he was an assistant professor in the Department of Computer Science at the University of Texas, Austin. He returned to the University of Maryland as an associate professor in 1981. From 1985 to 1994, he was the director of the University of Maryland Institute for Advanced Computer Studies, where he is currently a professor, and where he is also a professor and the chair of the Computer Science Department. He is a fellow of the IEEE.