

# Large-Scale Vehicle Detection, Indexing, and Search in Urban Surveillance Videos

Rogério Schmidt Feris, *Associate Member, IEEE*, Behjat Siddiquie, James Petterson, Yun Zhai, *Associate Member, IEEE*, Ankur Datta, Lisa M. Brown, *Senior Member, IEEE*, and Sharath Pankanti, *Fellow, IEEE*

**Abstract**—We present a novel approach for visual detection and attribute-based search of vehicles in crowded surveillance scenes. Large-scale processing is addressed along two dimensions: 1) large-scale indexing, where hundreds of billions of events need to be archived per month to enable effective search and 2) learning vehicle detectors with large-scale feature selection, using a feature pool containing millions of feature descriptors. Our method for vehicle detection also explicitly models occlusions and multiple vehicle types (e.g., buses, trucks, SUVs, cars), while requiring very few manual labeling. It runs quite efficiently at an average of 66 Hz on a conventional laptop computer. Once a vehicle is detected and tracked over the video, fine-grained attributes are extracted and ingested into a database to allow future search queries such as “Show me all blue trucks larger than 7 ft. length traveling at high speed northbound last Saturday, from 2 pm to 5 pm”. We perform a comprehensive quantitative analysis to validate our approach, showing its usefulness in realistic urban surveillance settings.

See demos at <http://rogerioferis.com/IEEEEMultimedia/>

**Index Terms**—Large-scale learning, large-scale video collections, vehicle search, video surveillance.

## I. INTRODUCTION

THE ability to automatically search for suspicious vehicles is extremely important in criminal investigation processes. In this paper, we present a complete system for automatic vehicle search in urban surveillance videos based on semantic attributes. Rather than relying on license plate recognition [1] or vehicle classification [2], [3], which may not be effective for low-resolution cameras or when the plate number is not available, we provide a complementary search framework which allows the user to search for vehicles based on attributes such as color, size, length, width, height, speed, direction of travel, date/time, and location. In particular our system is robust to challenging urban environments and supports large-scale data indexing.

Manuscript received January 26, 2011; revised June 20, 2011; accepted August 26, 2011. Date of publication October 06, 2011; date of current version January 18, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Alan F. Smeaton.

R. S. Feris, Y. Zhai, A. Datta, L. M. Brown, and S. Pankanti are with the IBM T. J. Watson Center, Hawthorne, NY 10532 USA (e-mail: rsferis@us.ibm.com; yunzhai@us.ibm.com; ankurd@us.ibm.com; lisabr@us.ibm.com; sharat@us.ibm.com).

B. Siddiquie is with the University of Maryland, Baltimore, MD 20742 USA (e-mail: behjat@cs.umd.edu).

J. Petterson is with the Australian National University and NICTA, Canberra 1435, Australia (e-mail: james.petterson@nicta.com.au).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TMM.2011.2170666

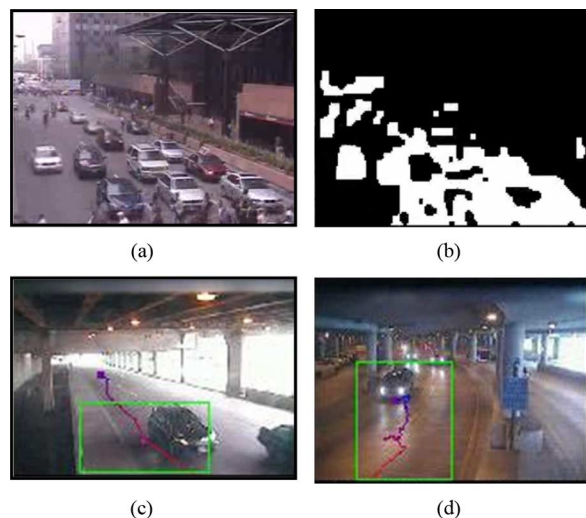


Fig. 1. Traditional methods based on background modeling fail to segment vehicles in common urban surveillance conditions. (a) Typical crowded scene. (b) Corresponding foreground blobs obtained through background subtraction. Note that groups of vehicles are clustered into the same blob. (c) Shadow effects. (d) Reflections. Note the vehicle bounding boxes are not precise, causing problems for subsequent modules such as vehicle color estimation.

Urban scenarios pose unique challenges for vehicle detection. High volumes of activity data, different weather conditions, crowded scenes, partial occlusions, lighting effects such as shadows and reflections, and many other factors cause serious issues in real system deployments, making the problem very challenging. Traditional methods based on background modeling [4], [5] generally fail under these difficult conditions, as illustrated in Fig. 1.

We propose a novel vehicle detection method that works reasonably well under these conditions. More specifically, we use a co-training technique where data is automatically captured in simple conditions (e.g., low activity levels) based on motion and shape cues and then used to train appearance-based detectors which work well in complex conditions (e.g., crowded scenes). Occlusions are explicitly handled by augmenting the training set with synthetically occluded vehicles generated by Poisson reconstruction from gradient fields. In order to handle multiple vehicle types (e.g., buses, trucks, compact cars), we use a deformable shape/aspect ratio sliding window. Finally, we train appearance-based vehicle detectors using large-scale feature selection, showing that the accuracy and sparsity of the classifiers are considerably improved by doing massively parallel feature selection over multiple feature planes.

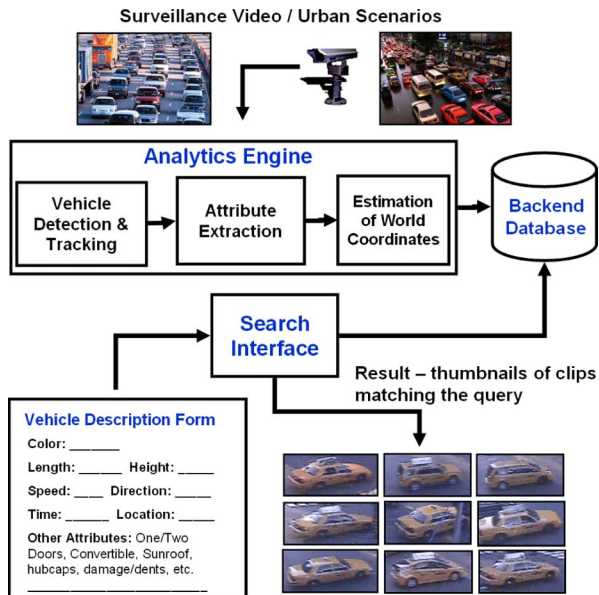


Fig. 2. System architecture. At the interface, the user specifies a set of vehicle attributes (vehicle description form) and the system retrieves the events matching the provided description. Our system is designed to handle high volumes of activity.

Once vehicles have been detected and tracked, attributes are extracted and ingested into a database. In particular, measurements such as speed, width, height, and length of vehicles are converted to world coordinates through a simple calibration process, thus allowing search across cameras without perspective issues. Fig. 2 shows a high-level overview of the architecture of our system.

The key contribution of our work is an end-to-end system for vehicle retrieval based on semantic attributes, including a robust detection/tracking approach for capturing vehicles in crowded scenes, while supporting large scale data indexing, where hundreds of billions of events need to be archived per month. The system interface and operation can be seen at: <http://rogerioferis.com/IEEEMultimedia/InterfaceAndOperation.wmv>.

## II. RELATED WORK

In the past few years, semantic attributes have received renewed attention in the computer vision community, providing effective high-level object representations for a variety of applications, including zero-shot recognition [6], [7], face verification [8], and people search [9]. Although semantic concepts and attributes have been inherently present in image retrieval, scene classification, and broadcast video search, we are not aware of any surveillance system capable of searching for vehicles based on their fine-grained attributes.

Most commercial surveillance systems rely on background modeling for detection of moving objects, in particular vehicles. However they fail to handle crowded scenes as multiple objects close to each other are often merged into a single motion blob. Environmental factors such as shadow effects, rain, snow, etc. also cause issues for object segmentation.

Various models and methods have been proposed for appearance-based object detection, in particular vehicle detection. Examples include the seminal work of Viola and Jones [10] and many extensions using different features, such as edgelets [11] and strip features [12], as well as different boosting algorithms like Real Adaboost and GentleBoost. Support vector machines with histograms of oriented gradients have also been a popular choice for object detection [13], [14]. In earlier work, Schneiderman and Kanade [15] showed good vehicle detection results using statistical learning of object parts. Although appearance-based detectors have achieved very good performance in challenging scenarios, they usually require tedious labeling of thousands of training samples to work well. In addition, most methods run below 15 frames per second in conventional machines, which is not desirable for large-scale surveillance systems requiring many video channels to be processed by a single machine.

Co-training and online learning methods [16], [17] alleviate the manual labeling issue, while constantly adapting the detector as new data comes in. A common limitation of these techniques is the inaccuracy in capturing online data to correctly update the classifier. Differently, our method uses a simple combination of motion and shape cues to capture a diversified set of vehicles during extended periods of time.

Several datasets have been proposed for learning and evaluation of vehicle detection algorithms. Examples include the UIUC [18] and USC [19] datasets as well as generic object recognition datasets which include cars as an object category, e.g., Caltech [20], MSRC, and the yearly Pascal VOC challenges [21]. However, these datasets mostly consist of images of vehicles restricted to frontal/rear and side poses and the number of images of cars is of the order of 1000, which in our opinion, is insufficient for capturing the entire degree of variation in the appearance of cars due to changes in pose, viewpoint, illumination, and scale. Our new dataset consists of around 1 million images of vehicles in real-world urban surveillance settings.

Methods for occlusion handling in object detection [22] generally rely on object part decomposition and modeling [14]. In our application, however, these methods are not well suited due to the low-resolution vehicle images. Video-based occlusion handling from the tracking perspective has been addressed by Senior *et al.* [23], but it assumes objects are initially far apart before the occlusion occurs.

Large-scale learning is an emerging research topic in computer vision. Recent methods have been proposed to deal with a large number of object classes [24] and large amounts of data. In contrast, our approach deals with large-scale feature selection, showing that a huge amount of local descriptors over multiple feature planes coupled with parallel machine learning algorithms can improve not only the detector accuracy, but also its efficiency at test time.

In our previous work [25], we have shown a preliminary version of our vehicle detection system. The present work proposes an enhanced detector approach which supports multiple vehicle types and additionally covers vehicle tracking, large-scale data indexing, and an application for attribute-based vehicle search.

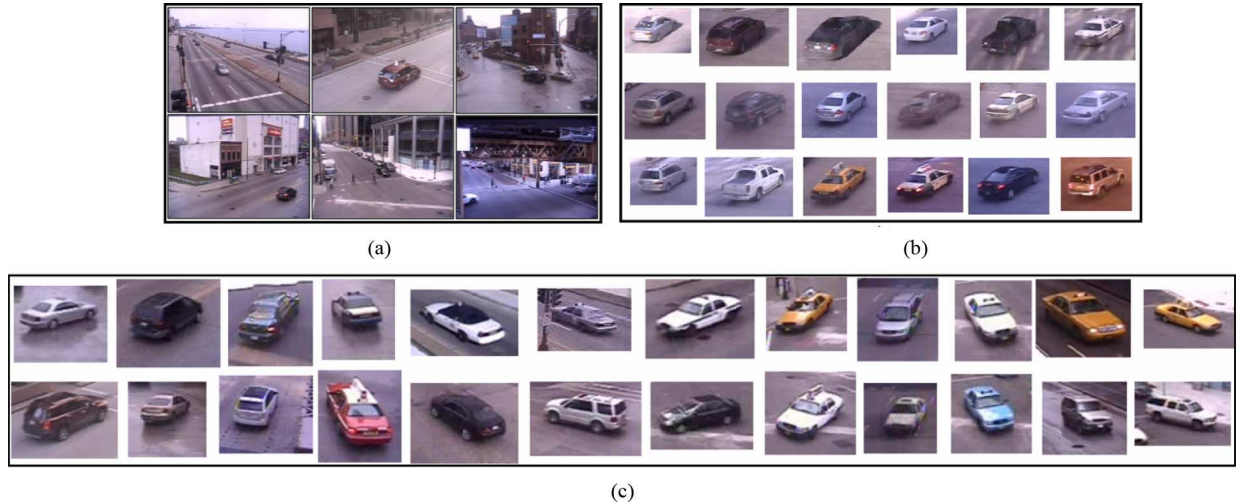


Fig. 3. (a) Sample shots from some of the traffic surveillance cameras. (b) Illumination variation in the dataset: Images of cars collected from a single camera view showing the illumination variation present in the dataset. (c) Vehicles categorized into 12 categories, according to their direction of motion  $\{0^\circ - 30^\circ, 30^\circ - 60^\circ, \dots, 330^\circ - 360^\circ\}$ , showing the diversity in pose of the cars in the dataset.

### III. VEHICLE DETECTION IN CROWDED SCENES

In this section, we describe our approach for automatic vehicle detection in urban scenes, including the training dataset formation, handling of occlusions and multiple vehicle types, large-scale detector learning, and vehicle detection benchmark.

#### A. Automatic Data Collection

Since it is costly to manually collect a large and diverse dataset consisting of cropped images of vehicles, we devised a simple procedure to semi-automatically collect images of vehicles in traffic videos, which we now describe.

We collected videos from about 30 traffic surveillance cameras [Fig. 3(a)], which constantly captured data over a period of several months, providing us with data in a variety of illumination and weather conditions. Furthermore, due to variations in the positions of these cameras with respect to the direction of traffic, there were large variations in poses of vehicles captured from different cameras.

In each camera-view, vehicles usually appear in 1–4 poses, as a traffic camera typically overlooks either a single road or an intersection. We collect data per camera for each vehicle pose independently by following the steps below:

- We manually define one or more regions-of-interest (ROI), which specify the regions of the image from where we want to capture vehicles.
- In each ROI, we perform background subtraction [5] to obtain the bounding boxes of foreground blobs at each video frame. We also obtain the associated motion direction of foreground blobs through standard optical flow computation.
- We collect vehicles by using a simple rule-based classifier which analyzes the shape and motion of foreground blobs at fixed time intervals. More specifically, in our implementation, we just check whether the *aspect ratio*, *size*, and *motion direction* of a particular foreground blob are within a pre-defined range of values that characterizes a vehicle. The range of values is obtained heuristically by taking the

maximum and minimum values of aspect ratio/size/motion direction from very few manually labeled exemplars (usually 1–4, depending on the number of vehicle poses in the scene).

This simple procedure enables us to collect a large number of images of vehicles, while requiring minimal supervision. Fig. 3(b) shows examples of training samples captured using a 10-h video (from 8 am to 6 pm) for a specific camera and specific vehicle pose. In this experiment, we were able to capture few thousands of samples without false alarms. Note that we have *many false negatives as we are conservative*, e.g., in this video, the classifier rejected many vehicle samples in crowded periods, or vehicles with long attached shadows, etc. However, we can see in Fig. 3(b) that we capture samples containing a huge amount of appearance variation—different lighting, weather conditions, vehicle models, etc. This is extremely important for training the appearance-based detector described in Section III-D. Fig. 4 shows another example of data collection for a specific vehicle pose in different time periods.

Using this data collection method, we were able to collect a dataset consisting of about 1 million images of vehicles. We categorized each vehicle into one of 12 different categories depending on its motion direction, to give a coarse estimate of its pose [Fig. 3(c)]. The 12 categories were chosen assuming that detectors will be robust to 30 degrees pose variation, but more categories could be considered. There is a wide variation in the scale and pose of vehicles collected from different cameras, even when they have the same motion direction. We notice very few false alarms in the data collection process—in rare cases, e.g., when a group of small objects have the same aspect ratio, size, and motion direction of a vehicle. These samples were manually pruned from our dataset.

#### B. Dealing With Occlusions

Although the algorithm described in the previous section can collect vehicle samples under significant variation of ap-



Fig. 4. Automatically collected data for a specific vehicle pose in different time periods. See <http://rogerioferis.com/IEEEMultimedia/SideView-1.wmv> for a video demo. For this specific sequence, we are able to capture more than a thousand cars without false alarms.

pearance, it fails to capture samples with partial occlusion. In this section, we show a fully automatic process to generate realistic partially occluded vehicle images. Adding images with occlusions to the training set makes the detector much more robust to crowded scenarios, as we will show later in our experiments.

Fig. 5 illustrates our algorithm. For a given vehicle image  $I_A$ , we randomly select another vehicle image  $I_B$  with its associated foreground mask  $M_B$ . We first dilate  $M_B$  to make sure the mask entirely contains the vehicle. Then we follow the steps below:

- Let  $I_{AB}$  be the image formed by pasting the vehicle of image  $I_B$  (i.e., the region defined by  $M_B$ ) at a random location of the bottom of image  $I_A$ .
  - Compute the intensity gradient:  $G(x, y) = \nabla I_{AB}(x, y)$ .
  - Let  $G'(x, y)$  be the modified gradient field obtained by adding zeros to  $G(x, y)$  along the border pixels of the foreground mask  $M_B$ .
  - Reconstruct image  $I'_{AB}$  which minimizes  $|\nabla I'_{AB} - G'|$ .
- Image reconstruction from gradients fields, an approximate invertibility problem, is still a very active research area. In  $R^2$ ,

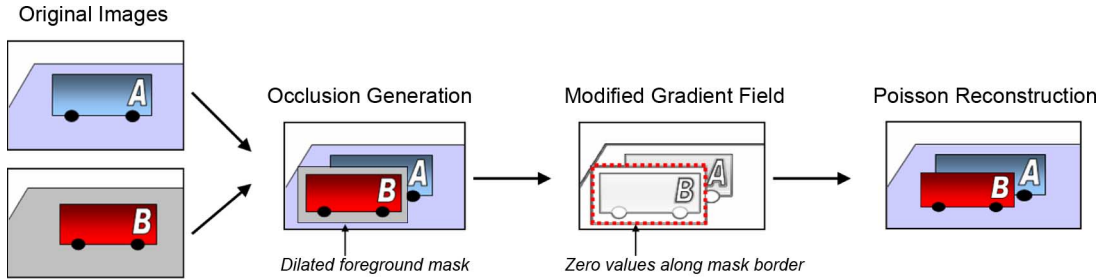


Fig. 5. Synthetic generation of occluded vehicles based on Poisson image reconstruction from gradient fields.



Fig. 6. Examples of realistic occluded vehicles generated by our algorithm. No user intervention is required.

a modified gradient vector field  $G'$  may not be integrable. We use one of the direct methods proposed by Fattal *et al.* [26]. The least square estimate of the original intensity function,  $I'_{AB}$ , so that  $G' \approx I_{AB}$ , can be obtained by solving the Poisson differential equation  $\nabla I'_{AB} = \text{div}G'$ , involving a Laplace and a divergence operator. We use the standard full multigrid method to solve the Laplace equation. We pad the images to square images of size the nearest power of two before applying the integration, and then crop the result image back to the original size.

The resultant image  $I'_{AB}$  is added to the collection of training samples, and this process is repeated so that many occluded vehicle samples are generated. Note that  $I'_{AB}$  provides a seamless blending of two vehicle images, even when they are captured under different lighting conditions. The superimposed image  $I_{AB}$  may contain pieces of the road and other artifacts due to noise in the foreground mask  $M_B$ . Since we modify the gradient along the border pixels of  $M_B$ , the resultant image  $I'_{AB}$  is much cleaner and realistic.

A similar algorithm is applied to generate images where the vehicle of image  $I_A$  (which is always positioned in the center of the image) is the occluder. Fig. 6 shows examples of realistic training samples generated by our method. The process is fully automatic.

It is important to note that we always have a vehicle in the center of the training image. The position of the other vehicle is constrained by just allowing it to be placed at random locations that will create partial occlusions, not full occlusions. The training images are further cropped to have a tighter bounding box around the vehicle in the center. Since at test time our detector is based on sliding windows, if we have one vehicle occluding another vehicle, then the detector will fire twice, i.e., we will have two bounding boxes (one for each vehicle).

### C. Dealing With Multiple Vehicle Types

We collected an additional set of images of trucks and buses for each camera view using our automated data collection tech-

nique with size and aspect ratio defined according to these vehicles. False alarms were manually pruned from the training set.

In order to deal with multiple vehicle types, we resize all training images so that they have the same aspect ratio. This way we can model only the appearance of all vehicle types irrespective of their sizes. The set of resized images therefore belongs to what we call a shape-free appearance space, since the information about aspect ratio is lost [Fig. 7(a)]. We use the term *shape-free appearance space* to denote the image space of objects with the same aspect ratio, without taking into account their detailed shape outlines as in active appearance models [27].

Note that we learn a single detector for multiple vehicle types (buses, trucks, and cars). Although different types of vehicles may differ in appearance, they also have many similar structures, which facilitates learning. As we will see in Section III-E, the aspect ratio of the sliding window is varied at test time to detect vehicles with different aspect ratio.

### D. Large-Scale Detector Learning

For each camera-view, we learn a specific vehicle detector for each vehicle pose using training samples collected automatically as described in the previous sections. Therefore, at test time, we have usually 1–4 detectors running per camera, which are interleaved across the video frames to improve frame rate. More specifically, rather than running all detectors in a single frame, we interleave them across the temporal domain, i.e., we run detector 1 in frame 1, detector 2 in frame 2, ... detector N in frame N, and then detector 1 in frame N+1, and so on.

The basis of our learning algorithm is the framework proposed by Viola and Jones [10]. It consists of a cascade of Adaboost classifiers, where the weak learners are simple thresholds over Haar-like features. Each stage of the cascade is tuned to minimize false negatives at the expense of a larger number of false positives—this allows fast inference by quickly discarding background images. Bootstrapping is also employed by selecting negatives examples where the previous stages have failed. For details, see [10].

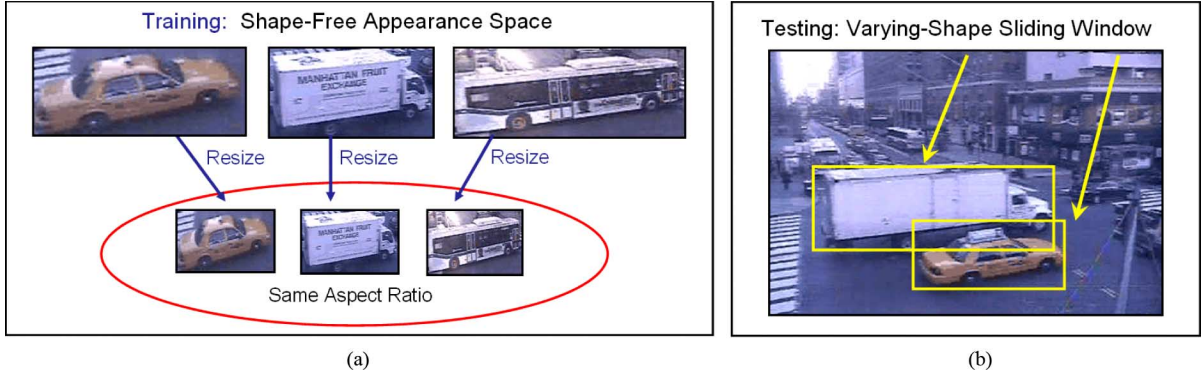


Fig. 7. (a) Training images containing multiple vehicle types are resized to have the same aspect ratio. (b) At test time, the aspect ratio of the sliding window is changed to enable the detection of various types of vehicles such as buses, trucks, and compact cars.

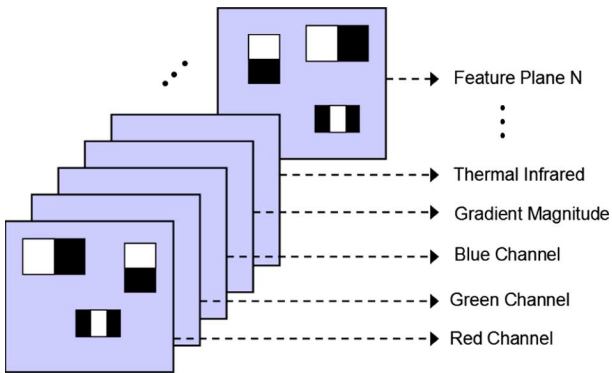


Fig. 8. Feature selection over multiple feature planes. A feature pool containing a huge set (order of millions) of feature configurations is generated.

The key novelty of our learning algorithm is the introduction of multiple feature planes in the feature selection process, as shown in Fig. 8. By considering feature planes such as red, green, and blue channels, gradient magnitude, local binary patterns, and many others, we allow the final detector to be much more powerful, combining Haar-like features of different modalities. In this framework, feature selection is performed over a pool containing a huge set (potentially millions) of feature configurations. This poses a serious problem in terms of training time, as even with a single feature plane and a few thousand images Adaboost training takes days on a standard desktop machine. Therefore, to deal with a huge pool of local feature descriptors as we are proposing, we need a way to parallelize training.

Adaboost is inherently sequential [28], making it difficult to scale in general, but in this particular setup, there is a simple solution: parallelization at the level of features. At each step during training, we have to compute a large number of features for all training images and select the one that better classifies the data; this can be done in parallel, with each CPU working on a subset of the features, and the amount of synchronization necessary is minimal: each CPU has to report only the best feature of its subset.

Additionally, at each stage, a set of negative patches has to be selected from the set of available negative images. The selected patches are the ones for which the current classifier fails. This is the most time-consuming activity in later stages

of the cascade training, taking hours even for a small training set. Parallelization can also be implemented here, with each CPU searching for negative patches in a different subset of the negative images. Again, the amount of time spent on synchronization<sup>1</sup> here is comparatively very small, allowing for an almost linear speed-up with the number of CPUs employed.

So far in our implementation, we have considered parallel feature selection over four color planes (gray-scale, red, green, and blue channels). As we will show in the experimental section, by adding color, we not only improve the robustness of the classifier but also get a sparser solution, with a smaller number of selected features. That, in turn, reduces computation time during inference. We are currently adding more feature planes (gradients and texture descriptors, multispectral planes, etc.), which should improve results even more.

### E. Running the Detectors

Given a test video frame, we use the traditional sliding window approach [10] to detect vehicles. The novel aspect is that we scan the image not only at multiple positions and scales, but also allow the search window to deform its shape, in particular its aspect ratio, as shown in Fig. 7(b). This enables the detection of multiple vehicle types, ranging from large trucks or buses to compact cars. Since our cameras are fixed, we apply this search scheme only on motion blobs obtained by background modeling to improve efficiency and accuracy.

Additionally, we interleave the classifiers across the video frames to obtain higher frame rates. Our vehicle detection system runs at more than 60 frames per second, being appropriate for real surveillance deployments which require many video channels to be processed per server.

### F. Vehicle Detection Benchmark

Rather than training a general, single vehicle detector using our large dataset, we are currently training multiple specific detectors for each camera. Without loss of generality, we choose one camera for our quantitative analysis. We collected a challenging test set from this specific surveillance camera containing 229 images and 374 vehicles of a single pose (side-view). The images were captured in different months, covering different weather conditions including sunny and

<sup>1</sup>We used message passing interface (MPI) in our implementation.

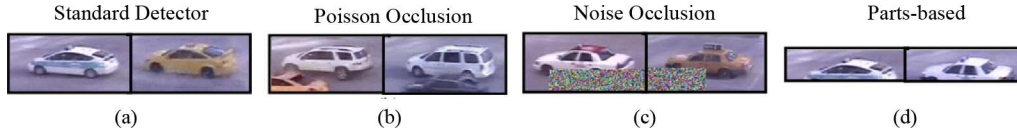


Fig. 9. Examples of training images used in different experiments.

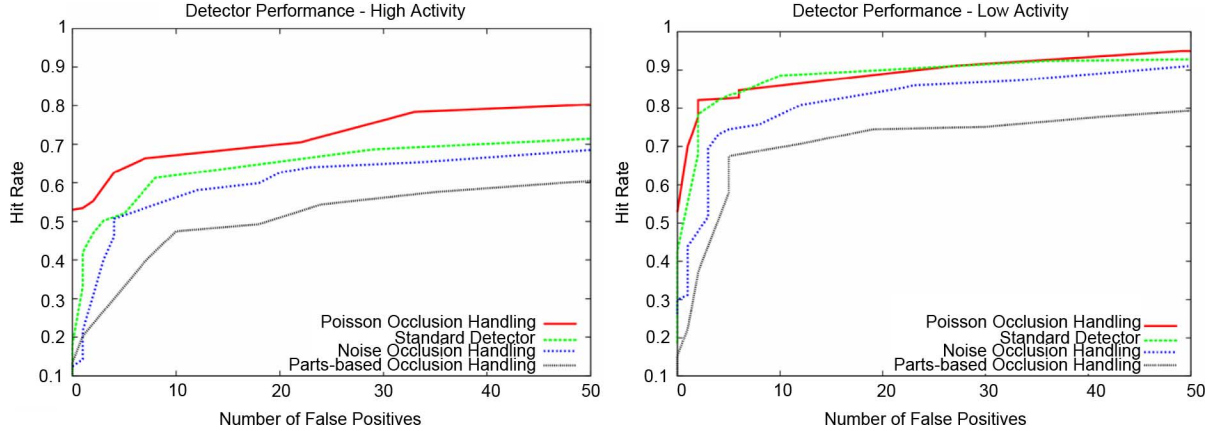


Fig. 10. Our occlusion handling approach significantly outperforms other methods in (left) high activity scenes, while having comparable performance to the standard detector in (right) low activity periods.

rainy days, different lighting effects, such as shadows and specularities, and different periods of time such as morning and evening. In addition, we split our test set into two groups: *high activity*, i.e., crowded scenes with many occlusions (104 images and 217 vehicles), and *low activity* (125 images and 157 vehicles).

We applied our data collection technique described in Section III-A to a 5-h (from 2 pm to 7 pm) video sequence of the same camera but in a different day/month of the period used to capture the test images. This way we could collect 1800 training samples automatically without any false alarms, which were then re-sized to  $26 \times 10$  resolution. A set of nearly 1000 negative images (non-vehicle data) were collected from the web and a cascaded Adaboost classifier based on Haar-like features was learned with a single gray-scale feature plane; later we will show the enhancement of learning with multiple planes. Fig. 9(a) shows a couple of examples of training samples of this standard detector. At test time, we apply the detector at different positions and scales, in the same sliding window scheme of [10]

**Occlusion Analysis.** In order to test our occlusion handling technique, we synthetically generated 1700 additional occluded vehicles using the technique described in Section III-B [Fig. 9(b)] and added them to the training set to create a new detector. For comparison, we also trained a detector using images with occluders consisting of random noise [Fig. 9(c)] and a part-based detector using images of the top part of the vehicles which are generally not occluded [Fig. 9(d)]. The ROC curves are shown in Fig. 10. The hit rate is the number of detected vehicles divided by the total number of vehicles. Note that our proposed approach significantly outperforms all other techniques in crowded scenes, while having comparable performance to the standard detector in low activity scenes, which is reasonable. The occlusion noise detector does not take

into account the appearance and structure of the occluder as in our approach. The part-based detector does not have good performance due to the low resolution of the vehicle images.

**Automatic versus Manual Labeling.** Fig. 11 shows a comparison of results obtained by training detectors using manual labels and labels collected automatically by our technique. We obtained 1800 manual labels from the same video sequence used in our quantitative analysis, considering crowded periods, partial occlusions, and a variety of lighting conditions. The performance of the detector trained using manual labels is superior in most operating points; however, our method is competitive, and in practice, it would be infeasible to manually annotate training images for each surveillance camera in large system deployments. We believe our method could come even closer to training with manual labels by automatically aligning the collected samples through, e.g., congealing. In crowded scenes, for low false positive rates, we reach better performance most likely because of the variety of artificially generated occlusions.

**Large-Scale Feature Selection.** We compared the standard Adaboost detector using one gray-scale feature plane with our approach using massively parallel feature selection over multiple feature planes. As of now, we used four planes (gray-scale, red, green, and blue channels). Fig. 12 shows the ROC curves and Table I shows the number of features selected in each plane. In addition to achieving improved accuracy, our large-scale feature selection scheme allows the final detector to be more sparse and therefore more efficient (see Fig. 14).

Sample detection results of our quantitative analysis can be seen in Fig. 13(a) and (b). We have also applied the same process (automatic data collection, occlusion generation, and detector learning) to other camera views, obtaining similar results. Qualitative results are shown in Fig. 13(c). Superior results can be obtained by applying our vehicle detector to video, by constraining the search process to foreground re-

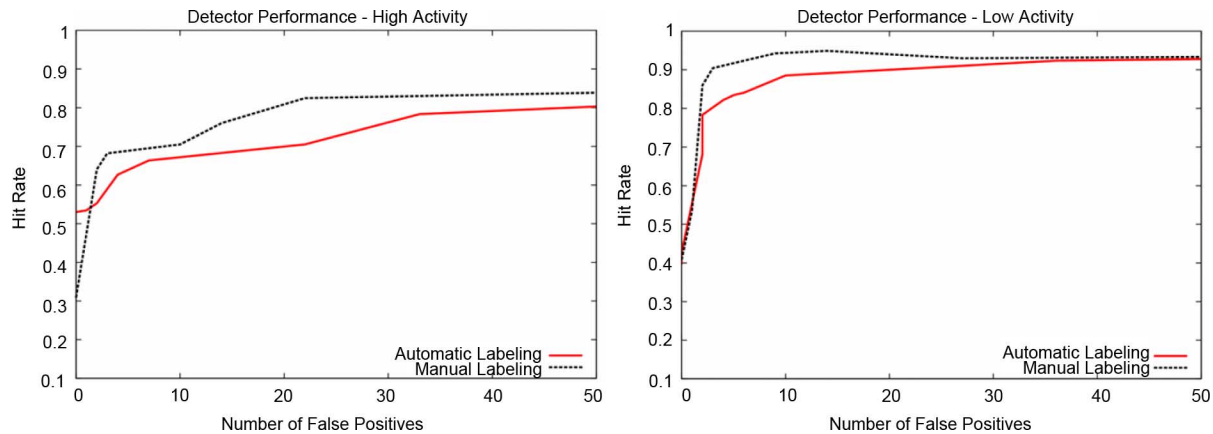


Fig. 11. Comparison of results obtained with automatic and manual labeling, in (left) high activity and (right) low activity periods. Although the performance of manual labeling is superior, it would be infeasible to manually annotate training images for each camera in real surveillance deployments.

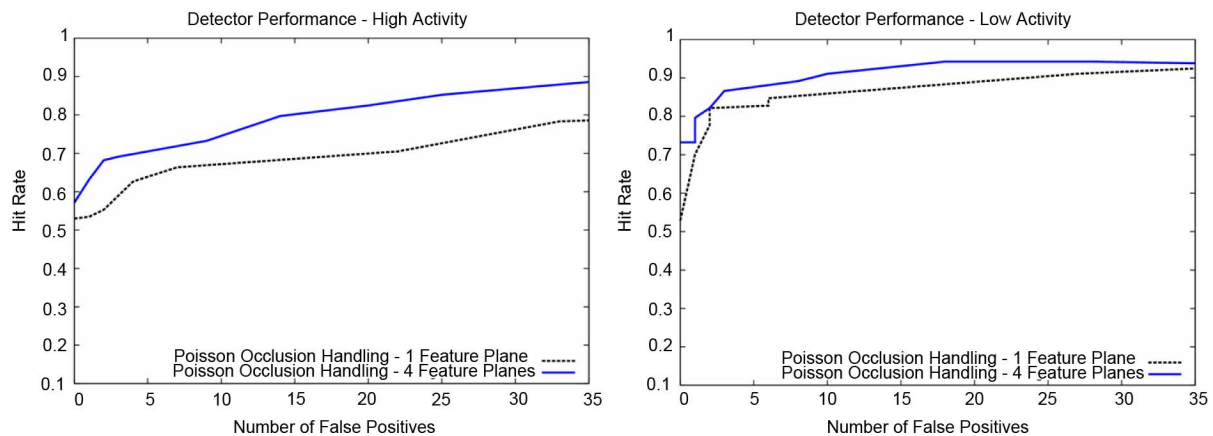


Fig. 12. Comparison of our approach using massively parallel feature selection over multiple planes with standard Adaboost detection using a single gray-scale feature plane, in (left) high activity and (right) low activity periods.

TABLE I  
NUMBER OF SELECTED FEATURES PER PLANE

plane	number of features	%
Grayscale	18	7.6
Red	82	34.6
Green	30	12.7
Blue	107	45.1

gions obtained through background modeling. We refer the reader to video demos at: <http://rogerioferis.com/IEEEEMultimedia/60HzVehicleDetectionCrowd.wmv> and <http://rogerioferis.com/IEEEEMultimedia/CrowdedDetectorDemo.wmv>. Our system runs quite efficiently, at an average of 66 Hz on a conventional laptop computer (2.3 GHz, 3 GB of RAM).

**Multiple Vehicle Types.** We qualitatively tested our method in a particular sequence that was obtained from New York Department of Transportation (DoT). It has low-frame rate and high volumes of activity. The video demo showing the detection of multiple vehicle types in crowded conditions can be seen at: <http://rogerioferis.com/IEEEEMultimedia/TruckCarsDetection-Crowd.wmv>.

**Discussion.** The reason why our method is robust to *environmental changes (shadows, rain, etc.)* is due to the fact that our data collection method captures many images under quite different lighting conditions, and also due to our rich feature pool.

The robustness to *crowds and partial occlusions* comes from our occlusion handling method. The ability to handle *multiple vehicle types* comes from our shape-free appearance learning and varying shape sliding window. The *efficiency* of our approach is due to the fact that we are learning detectors using samples of the same camera, which leads to a classifier with much less features than a generic vehicle detector. In addition, as we showed above, we obtain even more sparsity with large-scale feature selection. Failure cases include vehicle images occluded by more than 40% and other examples for which the detector is not able to generalize. Fig. 13(d) shows an example of a false alarm. The performance could be improved even more by collecting more training data over different days. As we run few pose-specific detectors per camera, we may not detect vehicles when they undergo poses not covered by the set of detectors, for example, when a car is turning. In our application, however, the detection output is used for indexing and to improve tracking, meaning that we need high precision but some false negatives can be tolerated. Finally, we note that our occlusion handling technique takes into account the appearance of vehicles occluding other vehicles, but not other types of occlusions, such as a vehicle occluded by a lamp post. This could be done by automatically collecting data from object tracks in low-activity periods as vehicles may get occluded.

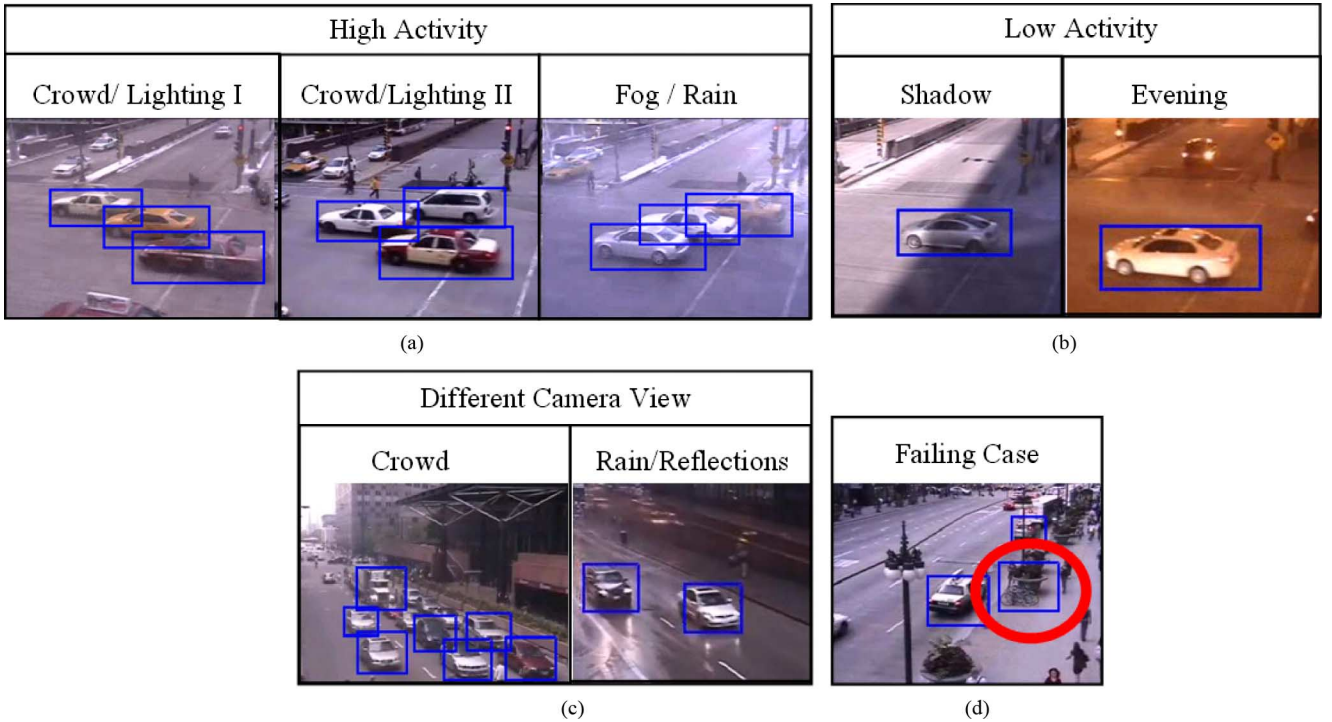


Fig. 13. Detection results. (a) High activity and (b) low activity samples used in our quantitative analysis (c) Different camera view. (d) Example of a false alarm.

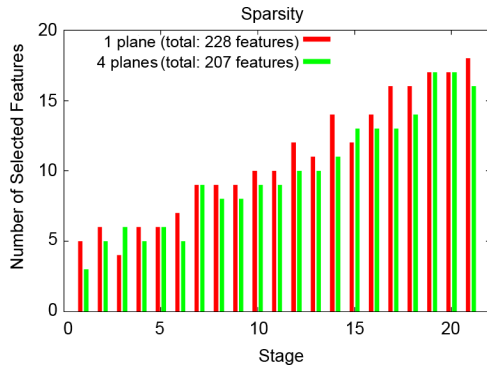


Fig. 14. Number of selected features at each stage of the classifier. Training with more features yields a sparser solution.

#### IV. VEHICLE TRACKING

In this section, we describe our vehicle tracking approach that utilizes the vehicle detection output described in the previous section. We use a *tracking-by-detection* approach [29], but also combine the information from object segmentation based on background modeling, so that we guarantee no events are missed.

**Tracking Framework.** The goal of our tracking framework is to combine multiple pieces of information to produce accurate motion estimates for objects in the scene. The input to our tracking framework consists of three pieces of information: detection results (previous section), background subtraction results [5], and previous time-step track estimates. Detection results gives us a rectangular box for every vehicle that has been detected in the current-frame. Background subtraction gives us a binary image of background and foreground. Note that while the results of detector and background subtraction often

overlap, they do not necessarily have to match. For instance, a vehicle may be in the foreground but missed by the detector or vice-versa where a static object has been detected but is not in the foreground. On the other hand, when the detection results do overlap on the background subtraction results, then we use the high-fidelity of the tracker to break apart the foreground blob into multiple regions as object hypothesis. For instance, if two cars are occluded, then background subtraction is going to generate one combined-blob for the two cars, whereas the detector may produce two individual detections. In such a case, we use the detection results to break the combined-blob into two smaller blobs. Finally, inclusion of previous time-step track estimates ensures that we make smooth associations of the current objects with the previous track estimates. Given these three pieces of information as input, we compute several cost-terms that measure the cost of association of a particular tracking hypothesis with a particular object measurement in the scene; a track hypothesis is composed of previous time-step track estimates, new tracks estimates, or merged track estimates. Given these cost of associations for every track hypothesis with object hypothesis, we use a greedy-version of the Hungarian algorithm [30] to solve data association.

**Association Costs.** Given a tracking hypothesis and an object measurement, the cost of association is measured in terms of three cost factors: centroid-distance, appearance, and size. Centroid-distance, as the names suggests, measures the distance between the centroid of a tracking hypothesis and the centroid of an object measurement. The greater the centroid-distance, the higher the cost of associating that particular tracking hypothesis and object measurement. Appearance cost measures the difference in appearance of a tracking hypothesis and an object measurement using sum-of-squared differences (SSD) in the overlapping area; the tracking hypothesis and the object measure-

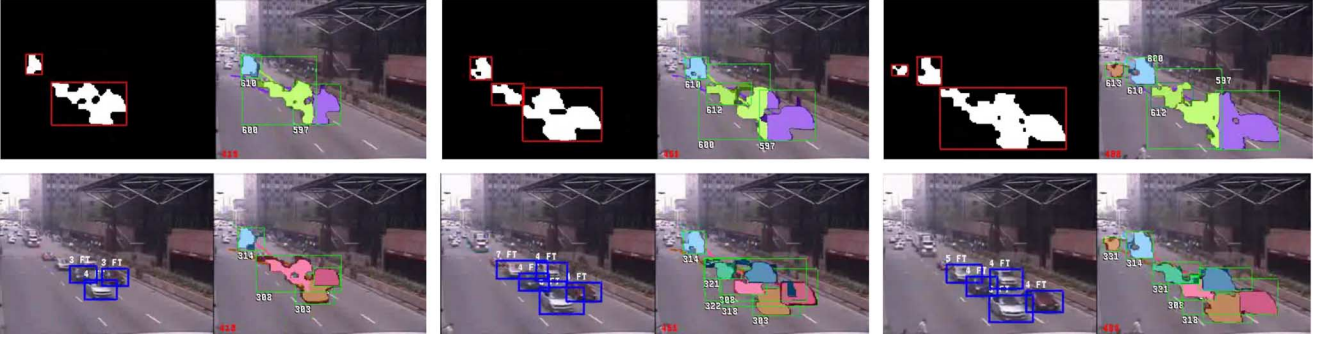


Fig. 15. Tracking Results: Top row—tracking results for three nonconsecutive frames of an image sequence where the left picture in each frame shows the result of background subtraction and the right picture shows tracking results using only background subtraction. Bottom row—left picture in each frame shows result of vehicle detection and the right picture in each frame shows tracking results using both the background subtraction and the detection results. Note that the inclusion of detection results allows us to perform tracking at a higher fidelity.

ment are aligned using their centroids. Higher appearance cost signifies dissimilar appearances and hence a higher cost of association. Finally, size cost measures the ratio between an object’s size and a tracking hypothesis size. The more this ratio deviates from 1.0, the higher the cost of association. The centroid-distance, appearance, and size costs for a particular track hypothesis and object measurements are then combined using a weighted summation, with weights manually tuned:

$$\begin{aligned} Cost(Tr, Obj) = & \alpha \times CentroidDistance(Tr, Obj) \\ & + \beta \times Appearance(Tr, Obj) \\ & + \gamma \times Size(Tr, Obj) \end{aligned} \quad (1)$$

where  $\alpha + \beta + \gamma = 1$ .  $Tr$  and  $Obj$  refers to track and object, respectively.

Weights for the association costs were determined using grid-search that was measured “pseudo-quantitatively”. A large group of exemplar video data was chosen, representing multiple possible scenarios including crowded and non-crowded scenes. A few frames per video were then labeled with object location along with a small part of its trajectory. Given such a labeled set of frames (around 1000 frames), we performed a coarse grid-search to optimize over the weights to come up with weights that generally work well across multiple scenarios.

**Data Association.** Given the cost of association of every combination of  $N$  tracking hypothesis and  $M$  object measurements, we can construct a  $N \times M$  cost of association matrix. The data association problem refers to the problem of assigning tracking hypothesis to object measurements such that the total cost of association is minimized. The Hungarian algorithm can be used to solve this problem optimally [30]. However, due to the real-time runtime requirements of video surveillance algorithms, we instead use a greedy-version of the algorithm [31]. Given the cost of association matrix, we select the tracking hypothesis and object measurement pair that has the least cost of association and delete that tracking hypothesis and the object measurement. This process of iteratively selecting the least cost of association tracking hypothesis and object measurement is continued until there are no more object measurements or tracking hypothesis left or if the cost of association rises above a particular pre-defined threshold.

**Qualitative Results.** Fig. 15 shows the results of tracking with and without the use of detection results. The top row in Fig. 15 shows tracking results on three nonconsecutive frames of an image sequence using only background subtraction and without the use of detection results. Note that many of the cars are clustered into a few large foreground objects that reduces the fidelity of the tracker. Fig. 15 bottom row shows the same three frames with the inclusion of the detection results. We can now break apart the large foreground blobs into individual detections and perform tracking at a higher fidelity.

As future work, we plan to carry out a quantitative tracking evaluation as offered in [32]. In terms of efficiency, we note that computational speed-ups could be obtained by exploiting the generic nature of motion of object categories involved in a typical surveillance scene. In fact, vehicles tend to move in straight lines with some amount of inertia in their motion. We believe that future tracking methods that can robustly incorporate higher-level reasoning based on generic motion of object categories can avoid a lot of computation overhead associated with re-detection.

## V. INDEXING AND SEARCH

In the previous sections, we described our algorithms for detection and tracking of vehicles in urban surveillance videos. This section covers the extraction of semantic attributes for each vehicle track, large-scale data indexing, and our application for attribute-based vehicle search.

In a criminal investigation process, eyewitnesses are typically asked by the police to fill out a *vehicle description form*, where they indicate the characteristics of the vehicle as seen at the moment of a suspicious activity or when a crime was committed. Those include direction of travel, license plate number, color, size, speed, body shape, wheel covers, decals, damages, presence of features such as sunroof, etc. Based on that description, the police manually scan the entire video archive looking for vehicles with similar characteristics. This process is tedious and time consuming.

Our goal is to automate this process by providing a vehicle search system based on semantic attributes. Our current implementation allows the user to search for vehicles based on color, size, length, width, height, speed, direction of travel, date/time,

and location, but many more attributes could be considered, including measurements from non-visual sensors.

## VI. ATTRIBUTE EXTRACTION/SEARCH

For each vehicle track, we extract a set of fine-grained attributes as described below and automatically ingest the attribute metadata into a backend database system through a web-based service-oriented architecture. SQL event search queries such as “Show me all blue trucks larger than 7 ft. length traveling at high speed northbound last Saturday, from 2 pm to 5 pm” can then be placed by the user through a web-based interface. The interface issues requests to a web server, where Java servlets receive the information and issue queries to the database backend. The results are then presented to the user. Thumbnails of the detected vehicles are displayed, and the user can click on them to view a video clip of the selected vehicle. The framework and software architecture of our vehicle search system is similar to the IBM Smart Surveillance Solution [33]. The search interface and a demonstration of our system in operation can be seen in the following video: <http://rogeriferis.com/IEEEMultimedia/InterfaceAndOperation.wmv>.

We currently extract the following metadata and attributes from vehicle tracks:

**Date, Time, and Location.** We store a timestamp indicating the beginning, end, and duration of a track. In addition, the information about the camera and its location in a map is stored, so that the user can search for events in a particular region of the city covered by certain cameras at a particular date/time.

**Direction of Travel.** We currently support search based on 12 motion directions. A motion direction histogram is built for each vehicle track, and the direction with larger number of votes is stored in the database.

**Dominant Color.** We extract the dominant color for each vehicle, allowing the user to search for vehicles based on six colors—black, white, red, green, blue, and yellow. The dominant color is computed by initially converting each input video frame into a bi-conic (hue, saturation, luminance) HSL space, and then quantizing the HSL space into six colors. This quantization is done by computing the hue angular cutoffs between the colors and, in a second stage, relabeling pixels as either white or black depending on whether they lie outside the lightness/saturation curve above or below the horizontal mid-plane. This is related to earlier work in color segmentation performed by Tseng and Chang [34]. A cumulative histogram with six bins in this quantized space is built over the vehicle images belonging to a specific track. The color corresponding to the bin which receives the majority of votes is then assigned as the dominant color.

**Vehicle Dimensions.** Our vehicle detection approach provides a precise bounding box and consequently the width and height in pixels for various types of vehicles. Pixel measurements, however, are sensitive to perspective, as a small car can look big if it is close to the camera and the other way round. We solve this issue by calibrating the scene and estimating the **width, height, and length** of vehicles in world coordinates, as described in the next section. We take the median value for each dimension over the entire vehicle track and ingest those values in the database.

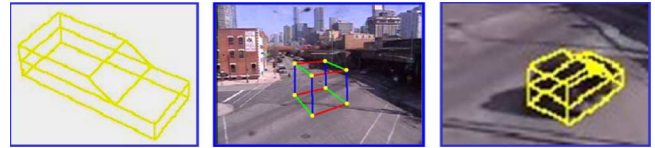


Fig. 16. Left: polygonal vehicle model. Center: the UI for using a 3-D cube to obtain the camera calibration matrix. Right: an example of fitted model to the observed vehicle using the calibration matrix.

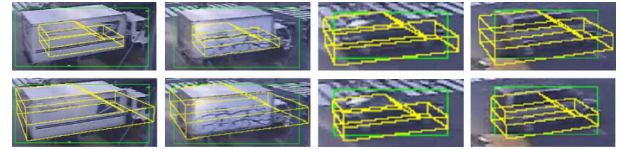


Fig. 17. Top Row: projected vehicles before 3-D scaling. Bottom Row: better fitted vehicle models after proper scaling.

**Speed.** Once we have the position of a particular vehicle in world coordinates at each video frame (see the next section), it is straightforward to compute its speed. We store the average speed of a vehicle track in the database.

Although our current implementation only considers the above attributes, more attributes such as wheel covers, body shape, presence of features such as sunroof, etc. could be considered.

### A. Estimation of Real-World Features

In many situations like traffic counting and large vehicle monitoring (e.g., trucks), knowing the actual 3-D characteristics of the vehicles such as width and length are important and critical. In this section, we present a real-world measuring method utilizing the camera calibration information. In particular, the 3-D dimensions of detected vehicles (width, height, and length) and their real speeds are estimated. To achieve this, a 3-D vehicle model (Fig. 16) is incorporated to represent the actual position and orientation of the vehicle in the real world.

The camera calibration is achieved by a manual specification process, where the user can define a 3-D cube, which is sitting on the ground plane and having edges with equal length (specified by the user, e.g., 20 ft.), through a convenient definition interface. The calibration matrix (a  $3 \times 4$  matrix) is then estimated using the least square fit method by matching the 3-D cube corners with their 2-D correspondences in the image plane. The user can also refine the calibration by visually viewing how well the vehicle model is projected to the image plane and fitted to an observed vehicle (Fig. 16). Note that in our application, only the projection matrix is needed. It is not necessary to know the exact parameters such as camera location, orientation, focal length, etc. Once the camera calibration matrix is obtained, it is used to match the 3-D vehicle model with the target object.

In order to correctly project the 3D vehicle model onto the 2D image such that its projection fits the detected vehicle, three things need to be known for the model: its location on the ground plane, orientation of heading direction, and the scale of the model. In our estimation process, the location is initialized as the intersection of the ground plane with the line that goes through the 2-D vehicle centroid and the camera center (using



Fig. 18. Color retrieval results. (a) Example results for a yellow car search. Note that for this particular scene, all taxi cabs have yellow color. (b) Quantitative results—confusion matrix and color distribution for the captured vehicles.



Fig. 19. Example of a search for large vehicles (larger than 18 ft.). Our vehicle detection method provides precise bounding boxes for multiple types of vehicles. Real vehicle dimensions in world coordinates are then obtained through a calibration and 3-D model fitting process.

backward projection). It is further refined once the other information is known. Assuming there are prior samples of the same detected vehicle from previous frames through the tracking process, its heading direction (denoted as  $\mathbf{v}_o$ ) is estimated as the motion vector between its current 3-D location and its previous location on the ground plane. The vehicle model is then rotated such that it aligns with this vector. If the vehicle is static, its previous heading direction will be used. For convenience, its perpendicular vector is denoted as  $\mathbf{v}_o^T$ .

There are many different types of motor vehicles, such as sedan, SUV, mini-van, medium truck, 18-wheeler, etc. Apparently, only knowing where the vehicle is located and to where it is heading is not sufficient to distinguish which type of vehicle it is. In this case, the scale of the vehicle must be estimated to make the correct inference. Here we provide a straightforward but robust method for approximating the vehicle scales. Given the bounding box of an observed vehicle in image, it can be denoted as  $BB_h(l, r, t, b)$ , where  $(l, r, t, b)$  are the left, right, top, and bottom coordinates of the bounding box, respectively. Similarly, the bounding box of the 2-D projected model can be also obtained and denoted as  $BB_m(l, r, t, b)$ . The aspect ratio difference between  $BB_H$  and  $BB_M$  in  $x$  and  $y$  directions are computed as  $S_x = r_h - l_h / r_m - l_m$  and  $S_y = t_h - b_h / t_m - b_m$ . Utilizing the backward projection technique again, we can find the 3-D vector  $\mathbf{v}_x$  on the ground plane whose 2-D projection aligns with the horizontal axis of the image, and similarly, the 3-D vector  $\mathbf{v}_y$  for the vertical axis of the image. Since these vectors are on the ground plane, their  $Y$  components are dropped. Assuming all the vectors are unit vectors, the scales in the length

$S_L$  and width  $S_W$  dimensions of the vehicle model can be estimated as follows:

$$\begin{cases} S_L = \alpha_L (S_x \|(\mathbf{v}_o \cdot \mathbf{v}_x)\| + S_y \|\mathbf{v}_o \cdot \mathbf{v}_y\|) \\ S_W = \alpha_W (S_x \|(\mathbf{v}_o^T \cdot \mathbf{v}_x)\| + S_y \|\mathbf{v}_o^T \cdot \mathbf{v}_y\|) \end{cases} \quad (2)$$

where  $\| * \|$  represents the absolute value of the vector dot product, and  $\alpha_L$  and  $\alpha_W$  are the normalization factors. Since the width and height of a vehicle usually are correlated, the scaling factor of model's height is defined as the same as  $S_W$ . Fig. 17 shows some example results for 3-D vehicle scaling. With the availability of vehicle's location, orientation, and scale, its 3-D features such as real length, width, height, and speed can be effectively estimated.

## VII. SEARCH EXPERIMENTS

Color retrieval was assessed on a specific video surveillance sequence from which we automatically captured 2550 vehicles. Fig. 18(a) shows an example of search for yellow vehicles. Note that in this case, all retrieved events are taxi cabs. Fig. 18(b) shows the confusion matrix and the initial distribution of vehicle colors. The percentage of images correctly classified is 96.76%. The average classification rate is 90.72%.

Evaluation of measurements such as width, height, length, and speed is not trivial as it is difficult to obtain the ground truth. So we provide a simple qualitative analysis by looking at search results associated with the query “search for vehicles with length larger than 18 ft.”, using the same video camera used for color retrieval over a period of 30 min. A total of 360 events were retrieved. Fig. 19 shows sample results for

Estimated events from vehicle traffic in a metro city	
Number of pole mounted street cameras in a large metro city	1000–5000
Number of events per camera. <b>Vehicle traffic only</b>	50,000/day/camera
Total number of events generated by the surveillance system/day	50 M–250 M
<b>Number of events archived per month</b>	<b>150B–750B</b>

Fig. 20. Estimated number of events in urban environments.

this search, including the timestamp and length information associated with each icon. By visually looking at the results, discriminating large cars from small cars (clearly above or below 18 ft.), we get 314 large vehicles out of the 360 events that were returned by the search, thus obtaining 87% classification accuracy. The classification errors usually come from false vehicle detections or wrong bounding box estimation. Note that traditional surveillance systems which offer object size search based on background/motion segmentation fail to handle crowded scenes as multiple small objects close to each other are merged and considered as a single large object. The video below shows a demonstration of our vehicle dimensions estimation in world coordinates: <http://rogerioferis.com/IEEE-Multimedia/WorldCoordinates.wmv>.

#### A. Large-Scale Data Indexing

A large urban surveillance system typically has the profile depicted in Fig. 20. Hundreds of billions of events need to be archived per month, which is a very challenging task. Next, we describe our approach to handle high volumes of real-time transactions and how we minimize the transaction/inquiry response time.

**Approach to High-Volume Scaling.** Given the event volume in Fig. 20, the data management layer (web application server and database) is required to support an ingestion rate of millions of events per day. Our system accomplishes this by using a combination of web application server clustering and database partitioning. In a clustered implementation of the web application server, a central node redirects traffic from cameras to secondary nodes to balance the load. As the number of cameras in the system increases, the system can be scaled by adding new web app server nodes that run on independent hardware, thus allowing scaling. At a secondary level, a similar philosophy is used to scale the database server using database partitioning where a single logical view of the database is supported by multiple instances of the software running on independent servers.

**Minimizing Transaction/Inquiry Response Time.** Most transaction and inquiry response delays can be attributed to either a non-optimized physical design or an insufficient disk storage layout for the data store. In a high volume environment, the characteristics of the metadata are highly important when designing the appropriate table and indexing mix for a highly scalable solution. We use the 20/80 rule to optimize access to the 20% of the metadata that supports 80% of the customer queries. It is also important to utilize a disk array, which has been configured to accommodate the amount of information that is being written from the data store. The following tasks need to be reviewed and implemented prior to installation: 1) Hardware disk controller with an adequate amount of cache to easily handle the known transaction volume; 2) total size of the data store with respect to the number of physical disks in

the storage array; 3) maximize the number of disk “spindles” to handle the current storage requirements; and 4) ability to add more storage space in the event that the number of cameras in scope increases. In addition to using the above design principles, we leverage the hardware-driven scalability of the database platform (DB2). This approach has resulted in event indexing systems that can handle hundreds to thousands of cameras on high-activity urban environments.

## VIII. CONCLUSION

The key contribution of our work is an end-to-end system for vehicle retrieval based on fine-grained attributes. We have presented a novel detection/tracking approach for capturing vehicles in challenging urban environments, being robust to crowded scenes and environmental factors. A comprehensive quantitative analysis was performed using real surveillance data.

Future work include 1) exploiting our vehicle dataset to learn a generic vehicle detector using millions of images; 2) adding more feature planes to generate a feature pool containing hundreds of millions of local features; and 3) developing large-scale online adaptation methods.

## REFERENCES

- [1] C. Anagnostopoulos, I. Anagnostopoulos, I. Psoroulas, V. Loumos, and E. Kayafas, “License plate recognition from still images and video sequences: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 377–391, Sep. 2008.
- [2] J. Prokaj and G. Medioni, “3D model based vehicle recognition,” in *Proc. WACV*, 2009.
- [3] X. Ma, W. Eric, and L. Grimson, “Edge-based rich representation for vehicle classification,” in *Proc. ICCV*, 2005.
- [4] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proc. CVPR*, 1998.
- [5] Y. Tian, M. Lu, and A. Hampapur, “Robust and efficient foreground analysis for real-time video surveillance,” in *Proc. CVPR*, 2005.
- [6] C. H. Lampert, H. Nickisch, and S. Harmeling, “Learning to detect unseen object classes by between-class attribute transfer,” in *Proc. CVPR*, 2009.
- [7] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, “Describing objects by their attributes,” in *Proc. CVPR*, 2009.
- [8] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, “Attribute and simile classifiers for face verification,” in *Proc. ICCV*, 2009.
- [9] D. A. Vaquero, R. S. Feris, D. Tran, L. Brown, and A. Hampapur, “Attribute-based people search in surveillance environments,” in *Proc. WACV*, 2009.
- [10] P. Viola and M. Jones, “Robust real-time object detection,” *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
- [11] B. Wu and R. Nevatia, “Cluster boosted tree classifier for multi-view, multi-pose object detection,” in *Proc. ICCV*, 2007.
- [12] W. Zheng and L. Liang, “Fast car detection using image strip features,” in *Proc. CVPR*, 2009.
- [13] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. CVPR*, 2005.
- [14] P. Felzenszwalb, R. B. Girshick, and D. McAllester, “Cascade object detection with deformable part models,” in *Proc. CVPR*, 2010.
- [15] H. Schneiderman and T. Kanade, “A statistical approach to 3D object detection applied to faces and cars,” in *Proc. CVPR*, 2000.
- [16] S. A. O. Javed and M. Shah, “Online detection and classification of moving objects using progressively improving detectors,” in *Proc. CVPR*, 2005.
- [17] P. Roth, H. Grabner, D. Skocaj, H. Bischof, and Leonardis, “On-line conservative learning for person detection,” in *Proc. PETS Workshop*, 2005.
- [18] S. Agarwal, A. Awan, and D. Roth, “Learning to detect objects in images via a sparse, part-based representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1475–1490, Nov. 2004.
- [19] C. Kuo and R. Nevatia, “Robust multi-view car detection using unsupervised sub-categorization,” in *Proc. WACV*, 2009.
- [20] G. Griffin, A. D. Holub, and P. Perona, The caltech-256, Caltech Tech. Rep., 2007.

- [21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. [Online]. Available: <http://www.pascalnetwork.org/challenges/VOC/voc2007/workshop/index.html>.
- [22] Y. Lin and T. Liu, "Fast object detection with occlusions," in *Proc. ECCV*, 2004.
- [23] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," *J. Image Vis. Comput.*, vol. 24, no. 11, pp. 1233–1243, Nov. 2006.
- [24] O. Russakovsky and L. Fei-Fei, "Attribute learning in large-scale datasets," in *Proc. ECCV 2010 Workshop on Parts and Attributes*, 2010.
- [25] R. Feris, J. Petterson, B. Siddiquie, L. Brown, and S. Pankanti, "Large-scale vehicle detection in challenging urban surveillance environments," in *Proc. WACV*, 2011.
- [26] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range," in *Proc. ACM SIGGRAPH*, 2002.
- [27] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685, Jun. 2001.
- [28] S. Merler, B. Caprile, and C. Furlanello, "Parallelizing adaboost by weights dynamics," *Computat. Statist. Data Anal.*, vol. 51, no. 5, pp. 2487–2498, 2007.
- [29] D. A. Forsyth, O. Arikan, L. Ikemoto, J. O'Brien, and D. Ramanan, "Computational studies of human motion: part 1, tracking and motion synthesis," *Found. Trends. Comput. Graph. Vis.*, vol. 1, no. 2-3, pp. 77–254, 2005.
- [30] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart.*, vol. 2, pp. 83–97, 1955.
- [31] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Online multi-person tracking-by-detection from a single, uncalibrated camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.
- [32] D. Sadlier and N. O'Connor, "Evaluation of a vehicle tracking system for multi-modal uav-captured video data," in *Proc. SPIE Defense and Security Conf.*, Orlando, FL, Apr. 2010.
- [33] A. Hampapur, L. Brown, R. S. Feris, A. Senior, C. Shu, Y. Tian, Y. Zhai, and M. Lu, "Searching surveillance video," in *Proc. AVSS*, 2007.
- [34] D.-C. Tseng and C.-H. Chang, "Color segmentation using ucs perceptual attributes," *Proc. Nat. Sci. Council: Part A*, vol. 18, pp. 305–314, 1994.



**Rogerio Schmidt Feris** (S'03–A'06) received the Ph.D. degree in computer science from the University of California, Santa Barbara.

He joined IBM T. J. Watson Research Center, Hawthorne, NY, in 2006, where he is currently a full-time Research Staff Member, responsible for research and development in the area of computer vision for intelligent video surveillance. He is an Affiliate Assistant Professor at the University of Washington, Seattle, and has also worked as an Adjunct Professor at the Computer Science Department

of Columbia University, New York. His research interests span a wide range of topics in computer vision, graphics, and machine learning, with emphasis on visual surveillance, intelligent user interfaces, and digital photography applications. He has published about 50 papers in peer-reviewed conference and journals, including publications in SIGGRAPH, ICCV, and CVPR.

Dr. Feris received several awards and distinctions, including an IBM Outstanding Innovation Award in 2011, IBM Emerging Leader in Multimedia in 2005, and Best Computer Science M.Sc. Thesis in Brazil-2nd Prize in 2002. He is one of the creators of the IBM Smart Surveillance Solution, which is an award-winning video surveillance system. He is the creator and general chair of the First International Workshop on Parts and Attributes, held in Greece as part of ECCV 2010.



**Behjat Siddiquie** received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Bombay, in 2006 and the M.S. degree in computer science from the University of Maryland, College Park, in 2009. He is currently pursuing the Ph.D. degree in the Department of Computer Science at the University of Maryland.

His research focuses on utilizing contextual information to improve the performance of computer vision tasks such as scene understanding and image retrieval. His other research interests include object recognition, activity recognition, and kernel-based learning methods.



**James Petterson** received the B.Sc. degree (with honors) in electrical engineering from Universidade Federal do Rio Grande do Sul (UFRGS), Brazil, in 1997. He is pursuing the Ph.D. degree at the Australian National University (ANU), Canberra.

After that, he worked as a Software Engineer in the telecommunications industry for more than ten years. In 2008, he took a position as a Research Engineer at National ICT Australia (NICTA). His research interests are in the general areas of machine learning and computer vision.



**Yun Zhai** (S'03–A'06) received the B.Sc. degree in computer science from the Bethune Cookman University, Daytona Beach, FL, in 2001, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL, in 2006.

He was a post-doctoral researcher in the IBM Thomas J. Watson Center, Hawthorne, NY, from 2006 to 2007, and he is now a senior research engineer in the Department of IBM Physical Security. His research interests are computer vision and multimedia signal processing. His recent work includes

temporal video segmentation, spatiotemporal visual attention modeling, and surveillance video analysis.



**Ankur Datta** received the B.Sc. degree in computer science with University Honors and Honors in the Major from University of Central Florida, Orlando, FL, in 2004 and the Ph.D. degree in robotics from the Carnegie Mellon University, Pittsburgh, PA, on an NSF Graduate Fellowship in 2010.

In 2010, he received the prestigious Far-Reaching Research (FRR) Grant from IBM Research to pursue research towards building the next-generation camera networks. He has published at numerous venues including TPAMI, ICCV, CVPR, ICPR, FG,

ICME, among others. His current research interests are in surveillance systems, human and object motion estimation, camera calibration, and structured learning.

Dr. Datta received the best paper award at the ICCV THEMIS workshop in 2009 for work on human motion estimation. He was also named a Computing Research Association (CRA) national finalist for his achievements in 2004.



**Lisa M. Brown** (SM'00) received the Ph.D. degree in computer science from Columbia University, New York, in 1995.

She wrote her thesis on medical image registration while working in the Computer Assisted Surgery Group at the IBM T.J. Watson Research Center, Hawthorne, NY. For the past 15 years, she has been a Research Staff Member at IBM Research. She worked for three years in the Education Group creating software which enables students to take measurements on images and videos. She is currently

in the Exploratory Computer Vision Group. She is well known for her ACM survey paper in image registration, which was extensively cited and translated into several languages. She has published extensively, been on several program committees for major conferences, been an invited speaker and panelist to various workshops and has filed more than 40 patents. Her primary interests

are in head tracking, head pose estimation, and more recently in object and color classification and performance evaluation of tracking in surveillance.

Dr. Brown is an editor for *Pattern Recognition*, the official journal of the Pattern Recognition Society. She was one of the inventors of the IBM Smart Surveillance Solutions (SVS) which is now an IBM Global Technology Services product offering and the winner of the 2008 North American Frost and Sullivan Award for Video Surveillance Software. She received the IBM Outstanding Innovation Achievement Award in 2007 and a Research Division Award for contributions to safety and physical security through video analytics in 2010.



**Sharath Pankanti** (S'93–A'96–SM'98–F'12) received the Ph.D. degree from Michigan State University, Ann Arbor.

He is currently the manager of the Exploratory Computer Vision Group at IBM T. J. Watson Research Center, Hawthorne, NY. He leads a number of safety, productivity, and security-focused projects involving biometrics, multi-sensor surveillance, driver assistance technologies that entail object/event modeling, detection, and recognition from information provided by static and moving sensors and

cameras. Many of these works are integrated into systems that have been rigorously evaluated in real-world applications. His research interests include performance metrics/evaluation, computer vision system designs for effective privacy, safety, security, productivity, and convenience. He has published about 70 publications in peer-reviewed conference/workshop proceedings and journals and has contributed to 20 inventions spanning biometrics, object detection, and recognition. He has co-edited the first comprehensive book on biometrics, *Biometrics: Personal Identification* (Norwell, MA: Kluwer, 1999), and co-authored *A Guide to Biometrics* (New York: Springer, 2004), which is being used in many undergraduate and graduate biometrics curricula.