Incremental Unsupervised Three-Dimensional Vehicle Model Learning From Video

Nirmalya Ghosh and Bir Bhanu, *Fellow, IEEE*

***Abstract*—In this paper, we present a new generic model-based approach for building 3-D models of vehicles from color video from a single uncalibrated traffic-surveillance camera. We propose a novel directional template method that uses trigonometric relations of the 2-D features and geometric relations of a single 3-D generic vehicle model to map 2-D features to 3-D in the face of projection and foreshortening effects. We use novel hierarchical structural similarity measures to evaluate these single-frame-based 3-D estimates with respect to the generic vehicle model. Using these similarities, we adopt a weighted clustering technique to build a 3-D model of the vehicle for the current frame. The 3-D features are then adaptively clustered again over the frame sequence to generate an incremental 3-D model of the vehicle. Results are shown for several simulated and real traffic videos in an uncontrolled setup. Finally, the results are evaluated by the same structural performance measure, underscoring the usefulness of incremental learning. The performance of the proposed method for several types of vehicles in two considerably different traffic spots is very promising to encourage its applicability in 3-D reconstruction of other rigid objects in video.**

***Index Terms*—Clustering, generic vehicle models, traffic surveillance, video-based 3-D modeling, 3-D vehicle modeling.**

I. Introduction

INTELLIGENT transportation systems (ITSs) and traffic-surveillance applications, such as vehicle tracking, automated highway tollbooths, autopilot vehicles, vehicle-type-based protected parking, etc., heavily depend on fair recognition/classification of moving vehicles. Present traffic-surveillance systems depend on license plate extraction [1], which is not robust to illumination variations. In addition, license plate recognition does not directly help in most of the applications other than detecting traffic-law-breaking cases. Hence, considerable research effort has been reported in vehicle detection, tracking, recognition, and classification, although predominantly in 2-D (see Table I). However, perspective projection in a 2-D image plane reduces classification accuracy, and the results are affected by occlusion and clutter. Reconstructed 3-D models of the vehicles are expected to improve performance in such traffic applications. Three-dimensional reconstruction of an object generally requires multiple views of the object, which can come from either a multiple-camera setup for static object or a static camera watching a moving object, where the views of the object change. In this paper, we use a static uncalibrated video camera watching moving vehicles. It provides different views in a partially redundant manner and has the potential for incremental 3-D modeling of vehicles from a video frame sequence. We have used a single *generic* 3-D vehicle model to evaluate correctness and to find weighted estimates of the incremental model over the frame sequences. Note that the focus of this paper is not vehicle tracking but 3-D model reconstruction of vehicles from video. There is no other work on incremental and unsupervised 3-D model reconstruction of moving vehicles from uncalibrated traffic color video.

In the next section, we provide an overview of the related work, the motivations for our work, and the contribution of this paper. In Section III, we begin with a novel template library approach for mapping 2-D features to 3-D model parameters for a single frame. We use these estimations in an adaptive-clustering-based incremental learning of the 3-D model of the vehicles from the video frame sequence and compute structure reliability scores that are used as both weights in 3-D model estimation and performance measures at different levels of abstractions. We end this section with the feature extraction technique used. In Section IV, we describe different traffic video data used and 3-D modeling results by the proposed approach. Finally, we conclude with a discussion of the results and future work in Section V.

Manuscript received February 19, 2009; revised July 10, 2009, September 29, 2009, and January 25, 2010; accepted March 30, 2010. Date of current version May 25, 2010. This work was supported in part by the National Science Foundation under Grant 0551741 and Grant 0905671. The Associate Editor for this paper was R. I. Hammoud.

N. Ghosh is with the Department of Pediatrics, Loma Linda University, Loma Linda, CA 92354 USA (e-mail: nirmalya@ee.ucr.edu).

B. Bhanu is with the Center for Research in Intelligent Systems, University of California, Riverside, CA 92521 USA (e-mail: bhanu@cris.ucr.edu).

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/TITS.2010.2047500

II. Related Work, Motivation, and Contribution

A. Related Work and Motivation

In a large proportion of the research on vehicle modeling and recognition [2], video/image data have been collected from a camera mounted on a vehicle, which is called the ego-vehicle, for autonomous navigation or autopilot applications [3], [4]. After moving vehicles are detected [5], stereovision is sometimes used to distinguish overlapping cars in 2-D [6]. For illumination invariance, infrared modality alone [7], [8], multimodality fusion [9], Gabor texture descriptors [10], and wavelet-based features [11] have been used. However, most of such works have been in 2-D (see Table I).

However, 2-D-based vehicle-surveillance systems suffer from the following: 1) Perspective projection may cause different vehicles to have similar silhouettes. For example, the frontal view of different vehicles may look quite similar in 2-D, whereas they have widely different 3-D structures. This

TABLE I
RELATED WORK ON VEHICLE MODELING AND RECOGNITION IN 2-D

Key methodology	Remarks
Temporal difference, minimum bounding rectangle (MBR), 2D modeling by linear segments, extended Kalman filter (EKF) tracking , mutual cooperation of recognition and tracking [3]	Video, follow the preceding vehicle
Principle component analysis (PCA) based modeling of different vehicles, projecting static-images to PCA subspace, matching, classification [14]	Video, classifying the succeeding vehicles
Detecting strong horizontal line as lower bounding of the frontal part of the vehicle, and tracking [15]	Video, detect and track the succeeding vehicle
Color, edge and motion information, temporal differences for moving edges, correlation for correspondence, motion computation, highway scene analysis [4]	Video, detect preceding vehicle, no occlusion
Neocognitron detection/recognition, preceding vehicle, supplementing partial features in occluded area [16]	Video, occlusion handling, only sedans/trucks
Laser-radar and video camera, detection of preceding vehicle and headway distance computation [9]	Laser/video, automated chasing, no make/model
Infrared (IR) sensors, local corner features from eigen-windows, prominent eigen-vectors, corner uniqueness by reliability, voting based recognition, hardware board for faster operation, indoor/outdoor setups [7, 8]	2D IR, make/model recognition, outdoor performance not good
Calibrated stereovision; finite occlusion/overlap templates, adaptable window for feature extraction [6]	Video, silhouette matching, finite templates
Gabor features (8 orientation/4 scales) for classification of four classes: cars, pickups, sedans, and vans [10]	Video, affected by shadows/illumination dir.
Frame subtraction; adaptive thresholds; heuristic shadow rejection; projection histograms, spatial wavelet 11]	Video, neural net classification, No occlusion
Parameterized model, corners/lines, street constraints, multi-layer-perceptron classifier [17]	Video, 2D recognition
Generic symmetric 3D vehicle model, deviation from mean shape analyzed by PCA, different weights in PCA space give different vehicle types, least square error (LSE) estimation, matching and classification [18]	Image, 2D sketches and single frame, edge-based approach
Aerial video, mosaic/multiscale features, trajectory graph, affine stabilization, traffic scene, no modeling [19]	Video, detection, tracking, activity recognition
Similar view, parametric alignment, iterative closest point (ICP) algorithm, 2D edge-based 2-class problem, learning, Fisher linear discriminants, Gibbs sampler, 2D classification [20]	Images, same size/pose across cameras
Quasi-rigid alignment, entropy-based flexible template matching, sequence-to-sequence matching, 3 principal directions, structural constraints [21]	Videos, pose change, tracking, not actually 3D
Probabilistic generative model with Gaussian PDFs of appearance, scale, shape, occlusion; entropy-based feature saliency, EM batch-mode learning [22]	Video, entire vehicle seen, high computational cost
Pose and model hypothesis: generation, tracking, verification in 2D [23]	Video, only scale variation
Model based tracking in monocular image sequences of road traffic scenes [24]	Video, Tracking only
Illumination-invariant tracking, graph-cut method, handles shadow and small occlusions [25]	Video, no model building

TABLE II
SELECTED RECENT 3-D MODEL BUILDING APPROACHES

Key methodology	Remarks
2 poses, mesh registration by tangent plane matching, iso-surface integration, toys [12]	3D range data, static background
Face, multi-viewpoints, 3D shape integration, intensity for texture integration [31]	3D range data , time intensive
Dense range data, 3D-to-3D registration, ICP algorithm, topological graph, color for only 2D-to-3D texture [32]	3D range data , historic buildings, slow
Buildings, corner matching by correlation/relaxation, finding camera parameters [13]	3D range data , time-consuming
Omnidirectional camera, panoramic, ICP registration, local features, degree-of-freedom (DoF) constraints [33]	Images, static scene model building
Range images, different overlapped viewpoints, point signature matching, brute search, head model [34]	3D range data, slow, temporal order unused
3D shape, parametric surfaces (S), 3D motion (M), 2D optical flow (R), $R = MS^T$, least-square factorization [28]	Video, box over carpet, too simplistic
Affine tracking of features, shallow structure, finding pose, refinement by pseudo-triangulation [26]	Images, rigid static data, monocular sequence
Aerial images, multiple views, missing data handling, feature template copying with hidden nodes, frequency-based probability distribution function (PDF) learning, Expandable Bayesian Network [29, 30]	Images, buildings, nearly top-view, avoids depth computation, no dynamics used
Learning based 3D vehicle tracking from video, 3D object appearance is learnt by eigen-representation [27]	Video, tracking, no 3D model building
Frontal car-crash, 2 orthogonal views, 3D deformation modeling from CAD model [35, 36]	Video, entire 3D model is not built
Close single static calibrated camera, Lucas-Kanade tracker (LKT), motion towards vanishing point, heuristic feature filtering, polynomial fitting to estimate 3D profile [37]	Video, specular reflection problems, slowly moving vehicles

creates ambiguity in classification. 2) Orientation changes are problematic. 3) Occlusion could be a major problem in 2-D, and it is better handled in 3-D than in 2-D. Hence, 3-D computer vision has long been a major area of research, and this requires 3-D model reconstruction from video. However, in most cases, researchers [12], [13] use range data collected by a laser sensor for 3-D reconstruction (see Table II).

Unfortunately, 3-D range cameras are far more costly to be used in rugged traffic environments. The availability of cheap video cameras and structure-from-motion methods have driven 3-D model building from video or image sequences (see Table II) in general [26]–[30]. With the exception of [27], the key shortcomings of the aforementioned work on vehicle modeling [26]–[30] are given as follows: 1) They work for detection or 2-D vehicle recognition, without building a 3-D model. A 3-D model can perform much better for occlusions and classifications. 2) They consider no adaptation and learning and are, hence, less robust to environmental variations. Even in [27], 3-D pose estimation and, then, 3-D–2-D transformation and matching are performed in 2-D for tracking over the video frames.

We find that little research has been done on 3-D vehicle model reconstruction from video and view-invariant vehicle recognition remains a challenging task. Current research on unsupervised learning of scale-invariant local features from 2-D structures [22] is in this direction but has yet to reconstruct a 3-D model. While detection [38], segmentation, and tracking [25], [39] have been addressed, 3-D model reconstruction has been ignored in most cases. Frontal 3-D deformation modeling in [35] and [36] uses 3-D computer-aided design (CAD) models (but does not build them from video) that are not always available for traffic vehicles. The close placement of the calibrated camera in [37] restricts the method only to constrained slowly moving vehicles and suffers from the reflections (visible due to

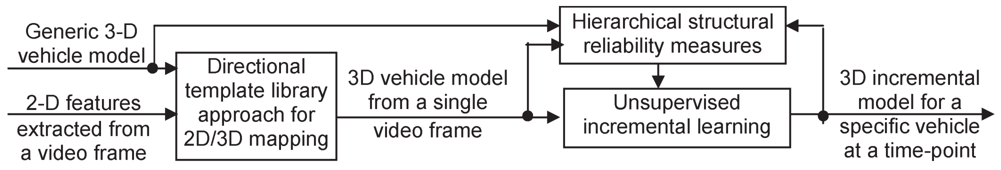
Generic 3-D vehicle model
2-D features extracted from a video frame
Directional template library approach for 2D/3D mapping
3D vehicle model from a single video frame
Hierarchical structural reliability measures
Unsupervised incremental learning
3D incremental model for a specific vehicle at a time-point

Fig. 1. Schematic of the technical approach.

closeness, and Lucas-Kanade Tracker (LKT) may be fooled). Similarly, classification by height profiles from radar [40] or multiscale texture-based key-component identification from a single view [41] is not cost effective, and it does not consider all the views. Rich information in the form of interframe view relations [42] in video data has not been utilized for 3-D model building.

General object-tracking strategies [43] do not always work well in real-world traffic scenarios. There are many model-based vehicle-tracking methods such as [44]–[49] for traffic-accident prediction or autonomous navigation. In some cases, known 3-D models and projected poses are used for vehicle localization. However, in all these methods, one needs reliable 3-D models of the vehicles, and often, the models used are too simple or too detailed to cover a wide variety of vehicles in traffic [50]. A standard 3-D reconstruction method is fitting a deformable 3-D vehicle model to an image sequence by edge matching in a 2-D image plane [50]. However, for this, we need a 2-D projection of a *complete* vehicle. This assumption of the *complete* vehicle being visible at different orientations is not true for traffic videos. The proposed method can take features from a partially visible vehicle (when entering the camera view) and start incremental 3-D reconstruction. A deformable model-based method will have to wait until the complete vehicle is visible. The bottom-up proposed approach does not start with a complete 3-D deformable model. Hence, this work is the first of its kind to use vehicular motion and view integration in video to incrementally reconstruct 3-D models of vehicles from uncalibrated static color video. It incrementally adds the model parts as they are gradually encountered in the video. The real applications need incremental 3-D reconstruction, with *partial* visibility of a vehicle in frames.

B. Contribution of This Paper

Our approach proposes incremental view integration and unsupervised incremental learning to produce reliable 3-D models of ground vehicles. The approach estimates frame-based 3-D features of a partially seen vehicle in the current frame, adaptively clusters the *same* features over the video frames seen until that point in time, and incrementally learns the parameters of a 3-D generic model for that particular vehicle. We use a cross-correlation-based 2-D tracking approach for feature extraction and correspondence and consider a wide variety of vehicles to validate the performance of our approach. The estimated 3-D model can be used for vehicle-type-based applications, such as automated toll stations, traffic-flow monitoring, and surveillance applications, e.g., the monitoring activity of a *particular* vehicle. The specific contributions of this paper are given as follows: 1) novel template-based matching to estimate 3-D orientation of a vehicle from a 2-D frame and to account foreshortening in projection, 2) incremental 3-D model building using correspondence across the frame sequence, 3) novel performance measures for 3-D modeling, and 4) experimental results using real video data of several vehicles.

III. Technical Approach

The proposed technical approach is summarized in Fig. 1. The extracted 2-D features of a vehicle and the parameters of a single 3-D generic vehicle model are used as inputs to a new template library-based approach to map 2-D features estimated to 3-D model parameters for a single video frame. Then, an unsupervised incremental learning method fuses the estimates from all the video frames up to the current point in time to obtain an incremental 3-D model of the vehicle. Hierarchical structural similarities are computed between the incremental model up to the last point in time and the 3-D model estimated using the current video frame only. These similarity measures (which are called reliabilities in this paper) are used in incremental learning as fusion weights and as performance measures at different abstraction levels. In Table III, we define all the mathematical symbols that are used in this paper.

A. Template Library Approach for 2-D/3-D Mapping for a Single Frame

In traffic scenarios, it is not cost effective to use laser cameras that can give 3-D depth information for 3-D model building of the vehicles. For using camera projection models, we need *calibrated cameras*. These are also not always possible because of the rugged traffic environment. Hence, we propose a novel template library approach to work with video data from a single static/fixed *uncalibrated* video camera to map the 2-D corners to 3-D to build the 3-D model of vehicles. The key assumptions are given as follows: 1) Vehicle 3-D surfaces are planar, and edge segments are linear. This is a valid assumption to some extent, and it has been utilized in feature extraction in Section III-D. 2) The vehicle can rotate in 3-D around the Z-axis only. This is generally true due to ground constraints of the streets. 3) There are different but constant 3-D to 2-D projection scales (S_x, S_y, S_z) for different 3-D directions. This is valid only when the distance from the camera to the vehicle is not too far for orthographic projection and not too close for perspective projection. The vehicle-to-camera distance should not vary too much for these constants to be fixed.

1) Generation of Template Library: Due to the perspective projection, linear distances are foreshortened, and 3-D solid angles between parts are nonlinearly mapped to their 2-D counterparts in the projected images. In addition, while working with uncalibrated traffic cameras, it is difficult to estimate the projection matrix. Furthermore, to add to the complexity, due to nonlinear perspective mapping and variations in the

TABLE III
LIST OF SYMBOLS USED IN THIS PAPER

Symbols	Description
D_{2D}, D_{3D}	Distances in 2D (pixels) / 3D (units correct to a scale factor)
[X_{img},Y_{img}]	Image plane coordinate system
[X_{3D}, Y_{3D}, Z_{3D}]	3D object centered coordinate (OCC) system
S = [S_x, S_y, S_z]	Relative scales to convert 2D distance to 3D, along OCC dir.
P = [α, β, γ]	Projections (2D angles) made by linear edge-segments concurrent at OCC origin, with X_{img} (Fig 2(b))
R, S_R, P_R	Azimuth or orientation of vehicle and OCC axes; corresponding scales S and projections P (Fig 2(a))
T_R	Template vector in Directional Template Library (DTL) for 2D to 3D mapping for azimuth angle R
Φ	Global motion direction in 2D on the image plane, w.r.t. X_{img}
V_{2D}(x, y)	2D vertex with coordinates (x,y) in [X_{img}, Y_{img}]
V_{3D}(X, Y, Z)	3D vertex; coordinates (X, Y, Z) in OCC [X_{3D}, Y_{3D}, Z_{3D}]
$V'_{3D}(X', Y', Z')$	Ground-truth from generic 3D model (i.e. incremental 3D model after previous video frame)
L_n, L_{2D},L_{3D}	Line L_n; 2D length L_{2D} (pixels) and 3D length L_{3D} (units)
L_{np}	A line non-parallel to all of the OCC axes [X_{3D}, Y_{3D}, Z_{3D}]
c2d	2D angle (degrees) subtended by L_{np} with Y_{img} on img. plane
b2d (see Fig 3)	2D angle (degrees) between L_{np} & Y_{3D} on $Y_{3D} - Z_{3D}$ plane
a2d	2D angle (degrees) between Y_{3D} and Z_{3D} on image plane
b3d (see Fig 3)	3D angle (degrees) between L_{np} & Y_{3D} on $Y_{3D} - Z_{3D}$ plane
$L_{3D,Y}$, $L_{3D,Z}$	Y & Z component of L_{3D} of L_{np} along Y_{3D} and Z_{3D} resp.
DP_{Ln}	3D Directional parameters of line L_n defined by terminal vertices V_{3D}(1) and V_{3D}(2)
LnCong	Line congruity in terms of line parallelism to [X_{3D}, Y_{3D}, Z_{3D}]
DirCong	Directional congruity matrix with each element containing LnCong values for a pair of vertices
E, φ, θ	LnAngErr(E): Angular error in line estimation, with ground-truth angle (φ) and estimated angle (θ)
C	LnCompRatio: ratio of ground-truth & estimated line lengths
D, v_{3D}, v'_{3D}	SubVdisp (D): disparity between ground-truth (v'_{3D}) location and estimated location of a sub-vertex (v_{3D})
Vrlb, subVrlb, Erlb, Mrlb	Reliabilities of a vertex (Vrlb), its corresponding sub-vertices (subVrlb), line or edge-segment (Erlb), and the incremental 3D model estimated (Mrlb)
DispW, rlbW	Weights for disparity (dispW) and reliability (rlbW)
μ1, σ1, μ, σ, σ'	Clustering mean & stdDev. before (μ1, σ1) & after (μ, σ) outlier rejection; normalized stdDev (σ').
F, Q	Exponential forgetting: F is either V_{3D}, or v_{3D}, or subVrlb, or Erlb with scale-factor for forgetting: Q
angSimTh	Angular similarity threshold to enforce line parallelism

distance/angle of the vehicle from the camera, the projection matrix needs to be updated. This is cumbersome and sometimes somewhat redundant, as will be shown in this paper.

In this work, we use a novel approach called "template library" to approximately estimate the 3-D-to-2-D nonlinear mapping relations in perspective projection. When assumption 3 is valid

$$D_{\mathrm{3D}} = S * D_{\mathrm{2D}} \tag{1}$$

holds true with different scale factors S for different line orientations. Multiplicative factor S converts the 2-D pixel length D_{2D} to 3-D distance D_{3D}.

Computing the projection matrix from multiple views of a vehicle [51] is not possible when the distance between the vehicle and the camera is not exactly known. Vehicular linear ridges between the 3-D surfaces are primarily along three principal 3-D directions (i.e., along three object-centered coordinate (OCC) axes) [21], [44]. This saves us from determining an infinite number of S's for an infinite number of possible line orientations, even for a particular azimuth of the vehicle. As *most* of the 3-D linear edge segments of vehicles are parallel to one of the coordinate axes in OCC, we just need three

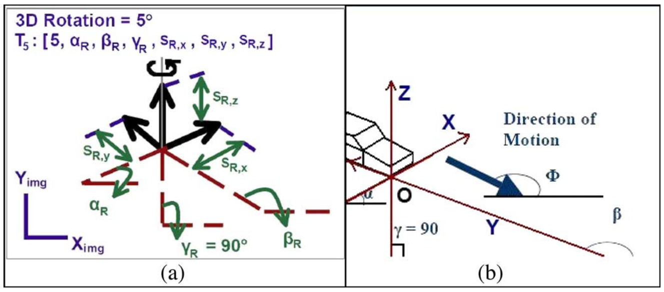

Fig. 2. (a) Sample template frame and corresponding template vector, with angles computed with respect to the image X-axis X_{img}. (b) OCC origin (O), corresponding axes, and motion direction (*best viewed in color*).

such constants (see Fig. 2) along each of the coordinate axes $[S_x, S_y, S_z]$. These scale factors vary for different possible azimuths, and we need to find $([S_{R,x}, S_{R,y}, S_{R,z}])$ for each of 360 possible azimuths (R). (Note that 180 of these are mirror reflections of the other 180.) Hence, a 3-D coordinate axes system, with *each 3-D axis of unit length*, is rotated around the Z-axis for 360 possible azimuths, and 360 templates are generated. For *each* frame, a template vector T_R is computed (offline) as in

$$T_R = [R, \alpha_R, \beta_R, \gamma_R, S_{R,x}, S_{R,y}, S_{R,z}]. \tag{2}$$

One sample frame with 5° orientation (azimuth) angle is shown in Fig. 2(a). The template library is the collection of 360 such vectors $(T_R; R = 1, 2, \ldots, 360)$ for 360 possible azimuths.

2) Finding 3-D Orientation and Projection Scales: For the current work, we assume that, for the entire video sequence, the frontal plane of the moving vehicle is visible, and we take the lower right vertex of this plane as the origin of the OCC. Note that this constraint can be relaxed if the OCC can dynamically be relocated to the left-hand lower corner of the rear plane of the vehicle or, in general, to any corner with three lines meeting at that point, where each line is along the principal directions of the vehicle 3-D edges. We decide the OCC origin for the first frame as the 2-D corner closest to the X_{img}-axis for all the video sequences considered. Then, we use matching in terms of the concurrent lines and the proximity over the consecutive frames to track the 2-D OCC origin.

The 2-D angles subtended by the three lines concurrent at the OCC origin in the image plane are computed as shown in Fig. 2(b). If the street is not too inclined (which is valid in most cases), the orientation constraint [by assumption 2] makes certain that the Z line in Fig. 2(b) is parallel to the image plane Y_{img}-axis. This is the OCC Z_{3D}-axis. To disambiguate between X_{3D} and Y_{3D} directions from the remaining two lines [X and Y lines in Fig. 2(b)] at the OCC origin, we use the global 2-D motion vector. This vector is decided by the following:

a) finding corner correspondence by proximity;
b) local motion vectors for the individual corners;
c) binning the motion angles in the histogram;
d) taking the most voted motion direction;
e) comparing with the last frame motion because the street constrains only smooth variation of the motion direction; if it varies significantly, take the last frame motion direction as the motion direction for the current frame.

We compute the interframe motion from 2-D feature displacement to find the motion angle Φ. As shown in Fig. 2(b), in general, vehicle-motion direction is close to the direction of the OCC Y_{3D}-axis. The 2-D line (at OCC) closely parallel to the motion direction is the OCC Y_{3D}-axis. The remaining one is the OCC X_{3D}-axis. Note that Φ and β are not always exactly the same because of possible rotations in the vehicular motion.

After getting the orientation $[\alpha\ \beta\ \gamma]$ of $[X_{\mathrm{3D}}\ Y_{\mathrm{3D}}\ Z_{\mathrm{3D}}]$ directions, as in Fig. 2(b), we find the best Euclidian match of $[\alpha, \beta, \gamma]$ over the corresponding columns [$[\alpha_R, \beta_R, \gamma_R]$ in (2)] of the template vectors in the library. The best match provides the estimate of the 3-D orientation of the vehicle R and estimated projection scales $[S_x, S_y, S_z]$ in principal directions, i.e., the OCC axis directions.

For finding the Euclidian match with the template vectors in the template library, all the similarities along different axis directions may not always be equally important. When the OCC line parallel to the Y_{3D}-axis is not visible, the Y_{3D}-axis direction is estimated by the motion direction, which may not be exactly in the same direction as of the actual Y_{3D}-axis (may be seen a few frames away), due to the rotation in the vehicular motion. Hence, we compute the weighted Euclidian distance with weight vector $W = [0.9\ 0.1\ 0.0]$. As the OCC Z_{3D}-axis is always very close to 90°, it has no importance in estimating the orientation. Similarly, when all OCC axis lines are visible, we take $W = [0.5\ 0.5\ 0.0]$. On the other hand, when, due to increased distance, orthogonal nature (orthographic projection) creeps in the projection of the lines and X_{3D}-axis line depth information is the most affected (due to being nearly parallel to the camera optical axis), to decide the template vector, we assign more weights on the Y_{3D} similarity, i.e., $W = [0.1\ 0.9\ 0.0]$.

3) 3-D Estimates: Vertices and Corresponding Lines: Two-dimensional/3-D location mapping starts by assigning the OCC 2-D origin to [0 0 0] and then uses the projection scales $[S_x, S_y, S_z]$ in different directions to map the 2-D line distances to 3-D distances along different propagation paths to get the 3-D locations of other corners. The key steps are in Algorithm 1.

Algorithm 1: 2-D/3-D mapping of features by the template library approach.

```
1. Enforce the parallelism between lines (see Section III-A3a).
2. Iterate for ($iter < maxIter$) OR (not all visible $V_{\mathrm{2D}}$ are mapped in 3-D).
3. For each of the 2-D visible corners ($V_{\mathrm{2D}} = [x, y]$)
    a. If it is already mapped to 3-D, consider the next visible 2-D corner.
    b. If it is the OCC origin in 2-D, assign 3-D location [0 0 0].
    c. If none of the other corners connected to $V_{\mathrm{2D}}$ are mapped to 3-D, then it cannot be mapped now, and it will be considered in later iterations when some of its connected vertices are mapped.
    d. If ($V_{\mathrm{2D}}$ is connected to a 2-D corner ($V1_{\mathrm{2D}} = [x1, y1]$) that is already mapped to 3-D) AND (the connected line is parallel to one OCC axis), then map $V_{\mathrm{2D}}$ by OCC parallelism (see Section III-A3b).
    e. If (none of the completely visible lines connected to $V_{\mathrm{2D}}$ is parallel to OCC axes) AND ($V_{\mathrm{2D}}$ is connected by an OCC nonparallel line to another corner $V1_{\mathrm{2D}}$ that has already been mapped to 3-D), then map $V_{\mathrm{2D}}$ by OCC nonparallelism approximations (see Section III-A3c).
4. If all the 2-D corners are not yet mapped to 3-D locations, then go to step 3 for the next iteration.
5. Modify the final 3-D mappings using structural congruities (see Section III-A3d).
```

a) Enforcing parallelism: For the propagation of estimated values and the symmetry of the vehicle rigid structures, we apply parallelism constraints. The lines that are nearly parallel to the OCC axis directions are enforced to be parallel. In this case, we use different angular similarity thresholds (*angSimTh*) for different vehicle-to-camera distances. The same *angSimTh* are used to enforce parallelism between pairs of lines that are not parallel to any of the OCC axes (i.e., the lines at the side of the windshield, back shield, or bonnet inclination).

b) 2-D/3-D mapping by lines parallel to the OCC axis: If 2-D corner V_{2D} ($[x, y]$) is connected to another 2-D corner $V1_{\mathrm{2D}}$ ($[x1, y1]$) that has already been mapped to its 3-D location $[X1, Y1, Z1]$ and the connecting line is parallel to one of the OCC axes, then we directly use the projection scales $[S_x, S_y, S_z]$ (see Section III-A2). We compute the 2-D length (L_{2D}) of the connecting line L_n in pixels and find to which OCC axis L_n is parallel. We use

if line L_n is parallel to $X_{\mathrm{3D}} : L_{\mathrm{3D}} = (L_{\mathrm{2D}}/S_x)$ and

$$V_{\mathrm{3D}} = \begin{cases} [X1 + L_{\mathrm{3D}} \quad Y1 \quad Z1], & \text{if } (x > x1) \\ [X1 - L_{\mathrm{3D}} \quad Y1 \quad Z1], & \text{if } (x < x1) \end{cases} \tag{3}$$

if line L_n is parallel to $Y_{\mathrm{3D}} : L_{\mathrm{3D}} = (L_{\mathrm{2D}}/S_y)$ and

$$V_{\mathrm{3D}} = \begin{cases} [X1 \quad Y1 + L_{\mathrm{3D}} \quad Z1], & \text{if } (x < x1) \\ [X1 \quad Y1 - L_{\mathrm{3D}} \quad Z1], & \text{if } (x > x1) \end{cases} \tag{4}$$

if line L_n is parallel to $Z_{\mathrm{3D}} : L_{\mathrm{3D}} = (L_{\mathrm{2D}}/S_z)$ and

$$V_{\mathrm{3D}} = \begin{cases} [X1 \quad Y1 \quad Z1 + L_{\mathrm{3D}}], & \text{if } (y > y1) \\ [X1 \quad Y1 \quad Z1 - L_{\mathrm{3D}}], & \text{if } (y < y1) \end{cases} \tag{5}$$

between directions of image plane axis and OCC 3-D axes to map V_{2D} into V_{3D}.

c) 2-D/3-D mapping for lines that are not parallel to the OCC axis: If none of the completely visible lines connected to corner V_{2D} ($[x, y]$) is parallel to any of the OCC axis AND 2-D corner V_{2D} is connected by an OCC nonparallel line (L_n) to another 2-D corner $V1_{\mathrm{2D}}$ ($[x1, y1]$) that has already been mapped to 3-D location $V1_{\mathrm{3D}}$ ($[X1, Y1, Z1]$), then we use geometric approximations and trigonometric relations to map V_{2D} to 3-D location V_{3D} ($[X, Y, Z]$) (see Fig. 3).

Note that all OCC nonparallel lines in common vehicles lie in one of the two possible Y_{3D}-Z_{3D} planes in OCC. Hence, the X_{3D}-axis coordinate of $V1_{\mathrm{3D}}$ and V_{3D} will remain the same. We find the 2-D slope angle (C_{2D} in Fig. 3) of line L_n made

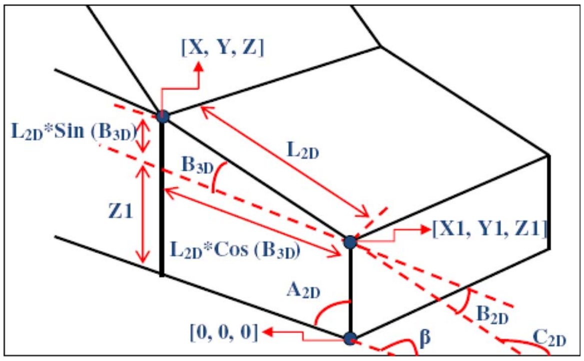

Fig. 3. Estimating 3-D vertex locations connected by edges that are not in parallel to any of the OCC axes (*best viewed in color*).

with the X_{img}-axis. As the projection of OCC $Y_{3\text{D}}$-axis makes a 2-D angle β (in Fig. 3) with the image plane X_{img}-axis, the 2-D angle $B_{2\text{D}}$ subtended by L_n with the OCC $Y_{3\text{D}}$-axis on the OCC $Y_{3\text{D}}$-$Z_{3\text{D}}$ plane is given by

$$\angle B_{2\text{D}} = \angle\beta - \angle C_{2\text{D}}. \tag{6}$$

Then, we find 2-D angle $(A_{2\text{D}})$ subtended by the OCC $Y_{3\text{D}}$-axis and OCC $Z_{3\text{D}}$-axis on the OCC $Y_{3\text{D}}$-$Z_{3\text{D}}$ plane by

$$\angle A_{2\text{D}} = \angle\beta - 90^\circ \quad \text{and} \\ \angle\ A_{2\text{D}}\ (\text{in 2D}) \ \Rightarrow\ \angle\ A_{3\text{D}}\ (\text{in 3D}) = 90^\circ. \tag{7}$$

Now, we postulate that the ratio of the two *concurrent* 3-D solid angles ($A_{3\text{D}}$ and $B_{3\text{D}}$) on the same 3-D plane will be equal to the ratio of the 2-D angles ($A_{2\text{D}}$ and $B_{2\text{D}}$) of the projection of those 3-D angles on the image plane. This is strictly true if the assumptions mentioned in Section III-A hold true. This implies that 2-D angles in the image plane can be mapped to their 3-D counterpart by linear interpolation. Thus, we find the 3-D counterpart $(B_{3\text{D}})$ of $B_{2\text{D}}$ (see Fig. 3) by

$$\text{On the same 3D plane, } \frac{A_{3\text{D}}}{B_{3\text{D}}} = \frac{A_{2\text{D}}}{B_{2\text{D}}} \Rightarrow \angle B_{3\text{D}} = \frac{\angle B_{2\text{D}}}{\angle A_{2\text{D}}} * 90^\circ. \tag{8}$$

If the length of the line L_n is $L_{2\text{D}}$, then the components of $L_{3\text{D}}$ along $Y_{3\text{D}}$ and $Z_{3\text{D}}$ ($L_{y,3\text{D}}$ and $L_{z,3\text{D}}$) in the OCC $Y_{3\text{D}}$-$Z_{3\text{D}}$ plane are computed by (9), shown at the bottom of the page, and then using assumption 2 under Section III-A and (10), shown at the bottom of the page, based on the 2-D relations between $V_{2\text{D}}$ $([x, y])$ and $V1_{2\text{D}}$ $([x1, y1])$ to map $V_{2\text{D}}$ to its 3-D location.

d) Modification by structural congruity: Finally, we apply the structural constraints from the generic 3-D model to modify the 3-D mapped coordinates so that, for all $X_{3\text{D}}$-axis, $Y_{3\text{D}}$-axis, and $Z_{3\text{D}}$-axis parallel lines, corners at both ends will be different only in the X, Y, and Z coordinates, respectively. This gradual correction is an iterative process.

B. Adaptive Clustering for Incremental Learning of Vehicle 3-D Model From Frame Sequence

1) 3-D Features: Using the template-library-based method described in Section III-A, we get the 3-D locations of the corners and the 2-D connectivity structure among the corners for a single video frame. Due to the limits of the view volume of the camera and the nonvisibility from self-occlusion, not all the corners and the corresponding connecting lines are completely seen in every frame. In addition, due to multiple possible paths to reach a particular corner from the OCC origin, we may obtain different estimates of the 3-D location from different paths. As more completely connected lines are seen for a corner, more different estimates are possible. We used an incremental clustering-based technique to integrate/fuse all this information to build a 3-D model of a vehicle. All the 3-D vertices are decoupled according to the connected lines, and each decoupled line terminal is called subvertex. The 3-D location of each subvertex is computed. From the 2-D corner connectivity structure, 3-D lines are automatically inferred. Thus, there are 3-D view-invariant features from a single frame.

a) Three-dimensional locations of the seen subvertices

$$V_{3\text{D}} = [X, Y, Z]. \tag{11}$$

b) Directional parameters of the completely seen edge segment, e.g., for edge segments L_n connecting $V1_{3\text{D}}$ $[X1, Y1, Z1]$ and $V2_{3\text{D}}[X2, Y2, Z2]$, i.e.,

$$\begin{aligned} DP_{L_n} &= V1_{3\text{D}} - V2_{3\text{D}} \\ &= [(X1 - X2) \quad (Y1 - Y2) \quad (Z1 - Z2)]. \end{aligned} \tag{12}$$

2) Incremental Learning Using Adaptive Clustering: Features estimated from a single frame are not very robust due to the approximations used. As more frames are considered,

$$L_{Y,3\text{D}} = \frac{\cos(B_{3\text{D}}) * L_{2\text{D}}}{S_y} \quad \text{and} \quad L_{Z,3\text{D}} = \frac{\sin(B_{3\text{D}}) * L_{2\text{D}}}{S_z} \tag{9}$$

$$V_{3\text{D}} = \begin{cases} [X1 \quad Y1 + L_{Y,3\text{D}} \quad Z1 + L_{Z,3\text{D}}], & \text{if } (x < x1)\&(y > y1) \\ [X1 \quad Y1 - L_{Y,3\text{D}} \quad Z1 - L_{Z,3\text{D}}], & \text{if } (x > x1)\&(y < y1) \\ [X1 \quad Y1 + L_{Y,3\text{D}} \quad Z1 - L_{Z,3\text{D}}], & \text{if } (x < x1)\&(y < y1) \\ [X1 \quad Y1 - L_{Y,3\text{D}} \quad Z1 + L_{Z,3\text{D}}], & \text{if } (x > x1)\&(y > y1) \end{cases} \tag{10}$$

incremental estimates are expected to become more reliable. As, in general, traffic video ground truth is not available (i.e., the 3-D model of any particular vehicle is *not* available), unsupervised learning has been adapted. Steps in the incremental unsupervised learning are shown in Algorithm 2. Each step is detailed in the succeeding sections.

```
Algorithm 2: Incremental learning of the 3-D model.
For each frame:
1. Extract corners and their connectivity structure for the current frame (see Section III-D). If any of the single-frame-based 3-D estimate is negative due to estimation propagation over lines with noisy edge points, discard that frame altogether.
2. Decide completeness of the vertices (see Section III-B2a).
3. For each corner
   a. Find its generic number in the 3-D generic model, and allot all its visible subvertices (see Section III-B2b).
   b. Clustering: Accumulate 3-D estimates for all visible subvertices over the frames seen so far (see Section III-B2c).
   c. Structural congruity: Compute $LnCong$, $DirCong$, $LnAngErr$ $(E)$, $LnCompRatio$ $(C)$, and $subVdisp$ $(D)$ (see Section III-B2d) of the 3-D estimates with respect to the 3-D generic model.
   d. Adaptation and outlier rejection: Fit 3-D Gaussian distribution for individual subvertices: Get mean $(\mu 1)$ and standard deviation $(\sigma 1)$. Remove points outside $(\mu 1 \pm 2\sigma 1)$ (see Section III-B2e).
   e. Unsupervised learning: Fit 3-D Gaussian for the remaining feature points, and get $(\mu, \sigma)$ (see Section III-B2f).
   f. Exponential forgetting: For each subvertex, the remaining estimates from step 3(d) are used with exponential forgetting to get the incremental estimates and reliabilities ($subVrlb$) (see Section III-B2g).
   g. Incrementally learning: Vertices are estimated by the weighted mean of subvertices and vertex reliabilities ($Vrlb$) by mean of corresponding subvertex reliability $subVrlb$ (see Section III-B2h).
4. Incrementally learned line reliabilities: Compute single-frame reliabilities, and then use exponential forgetting and incremental learning for the frames seen so far (see Section III-B2h).
5. Model reliability: Computed as a function of vertex reliability scores from step 3(g) (see Section III-B2i).
```

a) Deciding the completeness of the vertices: After discarding the frames with erroneous negative 3-D estimates of the vertices (from a single frame), we find the correspondence matrix $(Pt2LnCorr)$ between vertices and lines for the current frame. Then, the completeness of the vertices is found using the number of concurrent lines at that vertex. Complete vertices have more-than-one concurrent lines, whereas incomplete vertices can have only one.

b) Deciding proper subvertex allotment from generic model: In the generic 3-D vehicle model, for any 3-D generic vertex $(V'_{3\text{D}})$, line and vertex numberings follow two rules: 1) The line connecting the other vertex $(V1'_{3\text{D}})$ with the lowest generic number is counted first and then with next lowest generic number and so on, considering generic numbers in ascending order. 2) Vertices $(V'_{3\text{D}})$ are also considered in ascending order of the generic numbers. These rules specify unique subvertex numbers due to the corresponding lines and ensure that subvertex-level clustering is done with estimates that are computed in a similar manner.

For complete vertices, their actual generic model vertex numbers are found from the order of first appearance in the video sequence. For finding the exact subvertex number, current single-frame-based 3-D location estimates are saved and used in the following frames.

For each incomplete vertex $(V_{2\text{D}})$, there is only one line connected to it. We take four steps.

1) Find the other vertex $(V1_{2\text{D}})$, which must be a complete vertex, connected to $V_{2\text{D}}$.
2) From the generic 3-D vehicle model, find the generic vertex number of $V1_{3\text{D}}$ and its generic connectivity order for the other vertices to which $V1_{3\text{D}}$ is connected to in the generic 3-D model.
3) Among the connectivity list of $V1_{3\text{D}}$, discard those vertices that are already completely visible (as $V_{3\text{D}}$ is not visible yet), and among the rest, find the best match for $V_{3\text{D}}$ by the slope angle of the connecting line (by the method in Section III-A3c).
4) Store the estimate of the incomplete vertex at the correct subvertex variable based on the connectivity order of $V_{3\text{D}}$ in the generic 3-D vehicle model.

c) Clustering: During subvertex-level clustering, note that the estimates, like subvertex [X, Y, Z] locations, may vary, depending on the 2-D path taken during estimation propagation. Thus, different subvertices of the same vertex are estimated in different ways and are to be clustered individually. This is ensured by the method employed in Section III-B2b. As the single-frame-based 3-D location estimates are valid up to a scale factor, we normalize the coordinates to make the width of the vehicle fixed as this remains the same for nearly all vehicles due to fixed lane width in standard highways.

d) Structural congruity measurement: Structural congruity indices are the measures of similarity of the estimates to the corresponding counterpart in the generic 3-D model or the one incrementally learned until the last frame. They are used in incremental learning and estimation and as a performance measure at different levels of abstraction of the estimated 3-D model. For each complete 3-D vertex $(V_{3\text{D}})$, we consider each of its connected lines (L_n) and the other 3-D vertex $(V1_{3\text{D}})$ to which $V_{3\text{D}}$ is connected. Thus, we consider corresponding subvertices $v_{3\text{D}}$ and $v1_{3\text{D}}$, for $V_{3\text{D}}$ and $V1_{3\text{D}}$, respectively, for the line L_n to check the structural congruity. Two cases are possible: case 1, where both the terminal vertices are complete (when L_n is complete), and case 2, where only one of them is complete (when L_n is incomplete).

Case 1: Complete line L_n between V_{3D} and $V1_{3D}$:

1) Find the line congruity $(LnCong)$ defined by the parallelism of line L_n to different OCC axes, i.e.,

$$LnCong(L_n) = \begin{cases} 1, & \text{if } L_n \text{ is } X_{3D}\text{-parallel} \\ 2, & \text{if } L_n \text{ is } Y_{3D}\text{-parallel} \\ 3, & \text{if } L_n \text{ is } Z_{3D}\text{-parallel} \\ 4, & \text{if } L_n \text{ is nonparallel.} \end{cases} \tag{13}$$

2) Find the directional congruity matrix $(DirCong)$ defining the congruity of the line L_n between V_{3D} and $V1_{3D}$, i.e.,

$$DirCong(V_{3D}, V1_{3D}) = LnCong(L_n). \tag{14}$$

3) Find the subvertex disparity (D) that measures the weighted distance between the estimated V_{3D} and its generic counterpart V'_{3D}. The weights $(dispW)$ change [see (15), shown at the bottom of the page], depending on the importance of different OCC coordinates $([X\ Y\ Z])$ for that vertex and the completeness of the connected line (see the detailed discussion in Section III-C1).

4) Find the line completion ratio (C) of the line between V_{3D} and $V1_{3D}$ with respect to its ground-truth counterpart between V'_{3D} and $V1'_{3D}$ by (see also Section III-C1)

$$C = \|V_{3D} - V1_{3D}\| / \|V'_{3D} - V1'_{3D}\|. \tag{16}$$

5) Find the 3-D slope angle of the estimated line (θ) and then the angle error $(LnAngErr)$, using

$$E = abs(\theta - \varphi)/\varphi. \tag{17}$$

which measures the error between θ and its ground-truth counterpart (φ) in the generic model. For OCC nonparallel lines, the approximations described in Section III-A3c are followed.

Case 2: V_{3D} is a complete vertex, but $V1_{3D}$ is an incomplete one; then, the line L_n between them is an incomplete one. The generic vertex number of V_{3D} is known, and the generic vertex number of $V1_{3D}$ is decided by the similarity of the directional coefficients of 2-D line L_n and 2-D projections of the possible 3-D lines in generic or incremental model concurring on $V1_{3D}$. The directional/angular matching is done in the image plane. For two incomplete lines converging to the same corner, we will have the same generic vertex number, but correct 3-D subvertices are used in incremental clustering (see Algorithm 2). In this case, steps in the last paragraph are used with different $dispW$ in (15) to emphasize more dependence on those coordinates that are expected to be the same from the generic model constraints.

e) Adaptation and outlier rejection: For each of the complete 3-D vertices (V_{3D}) encountered in the current frame, we find the other vertices $V1_{3D}$ (complete or incomplete) connected to V_{3D}. Note that the incomplete vertices are automatically considered through the complete vertex to which it is connected. We accumulate corresponding subvertex 3-D estimations over the frames seen up to this point. We fit a 3-D Gaussian distribution with 95% confidence interval to get the mean $(\mu 1)$ and standard deviation $(\sigma 1)$ of the subvertex-level clusters. Adaptation is basically an outlier rejection step before the final unsupervised learning step. We discard the 3-D estimations outside the interval $(\mu 1 \pm 2^* \sigma 1)$.

f) Unsupervised learning: After discarding the outliers (in the last section), we fit another 3-D Gaussian distribution (with 95% confidence interval) with the remaining estimated locations of the same subvertex and find mean μ and standard deviation σ. Learned subvertex 3-D estimates are the corresponding means (μ's). Normalized standard deviation (σ') is the cluster variance that also measures the learning performance, i.e.,

$$\sigma' = \sigma / (1 + \|\mu\|). \tag{18}$$

Note that, as the incomplete lines gradually appear over the frames, the location estimates of the corresponding (incomplete) subvertex change. Hence, σ' is not significant to show cluster performance in these cases. Subvertex-level reliabilities $(subVrlb)$ for each vertex (V_{3D}) are computed by the following.

For complete vertices, with rlbW $= [1 \quad 1 \quad 2 \quad 5]/9$

$$subVrlb = rlbW * [C \quad 1/(1+D) \quad (1-E) \quad 1/(1+\sigma')]^T$$

and for incomplete vertices, with rlbW $= [1 \quad 1 \quad 5]/7$

$$subVrlb = rlbW * [C \quad 1/(1+D) \quad (1-E)]^T \tag{19}$$

with different reliability weights $(rlbW)$ for complete and incomplete vertices (as detailed in Section III-C1).

g) Exponential forgetting: It is noteworthy that, although we are estimating a rigid 3-D model of the vehicle in videos, the estimates from different frames are not the same due to different noise levels and different estimation errors due to approximations in Section III-A. For the incrementally learned estimate of the 3-D vertex locations of the model, exponential forgetting has been applied on the final cluster, as the feature points seen long before in a frame are less relevant for estimates in the current frame.

$$D = (V_{3D} - V'_{3D}) * dispW * (V_{3D} - V'_{3D}) / \|V_{3D} - V'_{3D}\|, \quad \text{where}$$

$$dispW = \begin{cases} L_n\text{is complete,} & L_n\text{is incomplete} & \\ [1 \quad 3 \quad 3]/7 & [1 \quad 5 \quad 5]/11, & \text{if } L_n \text{ is } X_{3D}\text{-parallel} \\ [3 \quad 1 \quad 3]/7 & [5 \quad 1 \quad 5]/11, & \text{if } L_n \text{ is } Y_{3D}\text{-parallel} \\ [3 \quad 3 \quad 1]/7 & [5 \quad 5 \quad 1]/11, & \text{if } L_n \text{ is } Z_{3D}\text{-parallel} \\ [5 \quad 3 \quad 3]/11 & [5 \quad 1 \quad 1]/7, & \text{if } L_n \text{ is nonparallel} \end{cases} \tag{15}$$

Individual incremental 3-D estimates of a visible subvertex (v_{3D}) and its corresponding incremental subvertex reliability values $(subVrlb)$ are computed using (20), shown at the bottom of the page, with corresponding entries across the frames. Q is the exponential forgetting factor. We use the incremental estimates of V_{3D} and $subVrlb$ for incremental 3-D model reconstructions.

h) Incremental learning of vertices and lines: Using the v_{3D} and $subVrlb$ from (20), we compute the incremental estimates of the vertices (V_{3D}) by (21), shown at the bottom of the page, and the corresponding vertex reliabilities $(Vrlb)$ by (22), shown at the bottom of the page.

Note that, once we have the estimates of the 3-D locations of the vertices and the 2-D connectivity matrix $(Pt2LnCorr)$ for the current frame, we have the 3-D lines in the current frame. Line reliabilities for an individual frame are found by (23), shown at the bottom of the page, using the structural congruity measures and reliabilities of the connected subvertices. However, different reliability weights $(rlbW)$ are used (explained in Section III-C3) based on the completeness of the line itself and completeness of the connected vertices. Then, we apply the exponential forgetting principle in (20) to get the incremental line reliabilities $(Erlb)$.

i) Incremental 3-D model of the vehicle: With the incremental estimates of the 3-D locations of the visible vertices of the vehicle and the 2-D vertex connectivity matrix $(Pt2LnCorr)$ from the 2-D features, we reconstruct the 3-D model of the vehicle. For performance verification, we define a unified reliability value of the 3-D model $(Mrlb)$, which is the mean of the vertex reliabilities, i.e.,

$$Mrlb = \sum_{i:\text{all visible vertices}} Vrlb_i \Big/ \sum_{i:\text{all visible vertices}} 1. \tag{24}$$

C. Reliability Scores as Performance Measures

Reliability scores are structural congruity measures of the estimates of 3-D corner locations and the lines connecting between them, with respect to the generic 3-D vehicle model. These measures serve two purposes in this work: 1) They act as dynamically adaptive weights for estimates of the 3-D model parameters incrementally modified at different abstraction levels (see Sections III-B2d–III-B2i), and 2) they act as performance measures at different finer levels to evaluate the estimated 3-D model at any time. Reliability scores have been measured at different abstraction levels, as follows.

1) Subvertex Reliability: Four factors govern the subvertex reliability.

a) **Normalized standard deviation** $(\boldsymbol{\sigma}')$: This is measured by (18) to quantify the divergence of the cluster used for unsupervised learning (see Sections III-B2f). In a way, this factor shows some idea of congruency in the data itself and the subvertex-level learning performance.

b) **Subvertex location disparity** (D): This measures (15) the weighted disparity of the estimated and generic locations of a vertex. The weight $(dispW)$ changes for

Incremental estimate of $F(V_{3D}$ or v_{3D} or $subVrlb$ or $Erlb)$ at frame t :

$$F(t) = \frac{\sum_{fr=1}^{t} e^{-Q(t-fr)} * k(fr) * F(fr)}{\sum_{fr=1}^{t} e^{-(t-fr)} * k(fr)} \quad \begin{array}{l} \text{where} : Q = \text{scale factor for forgetting} \\ k(fr) = \begin{cases} 0, & \text{if } F(fr) \text{ is outlier} \\ 1, & \text{otherwise} \end{cases} \\ \text{and} \quad fr = 0, 1, \ldots, N \text{ (total frames)} \end{array} \tag{20}$$

$$V_{3D} = \sum_{i:\text{visible subvertices}} subVrlb_i * v_{3D}(i) \Big/ \sum_{i:\text{visible subvertices}} subVrlb_i \tag{21}$$

$$Vrlb = \sum_{i:\text{visible subvertices}} subVrlb_i \Big/ \sum_{i:\text{visible subvertices}} 1 \tag{22}$$

$$\begin{aligned} Erlb &= rlbW * [C \quad (1-E) \quad 1/(1+D_1) \quad 1/(1+D_2) \quad subVrlb_1 \quad subVrlb_2]^T \\ \text{where}, rlbW &= \begin{cases} [2 \;\; 4 \;\; 2 \;\; 1 \;\; 4 \;\; 2]/15, & \text{if } V1_{3D} \text{ is complete}, V2_{3D} \text{ is incomplete} \\ [2 \;\; 4 \;\; 1 \;\; 2 \;\; 2 \;\; 4]/15, & \text{if } V1_{3D} \text{ is incomplete}, V2_{3D} \text{ is complete} \\ [1 \;\; 4 \;\; 2 \;\; 2 \;\; 4 \;\; 4]/17, & \text{if } V1_{3D} \text{ and } V2_{3D} \text{ are both complete} \end{cases} \end{aligned} \tag{23}$$

different cases. For example, with complete X_{3D}-parallel lines, the Y and Z coordinates of the estimated and the generic subvertex are expected to be more similar than the X coordinate, and hence, ($dispW = [1\ 3\ 3]/7$) is selected. For incomplete X_{3D}-parallel lines, ($dispW = [1\ 5\ 5]/11$) emphasizes less congruency in X coordinates of the estimated vertex to that in the generic model and, hence, less *relative* weight for X.

c) **Line completion ratio** (C): This is computed by (16). As the line lengths are unique for different vehicles, we adapt the ground-truth line length [denominator in (16)] from the incrementally built 3-D model at that point in time.

d) **Line angle error** (E): In (17), we gradually adapt the ground-truth line angle (φ) with that in the incrementally built 3-D model to accommodate uniqueness of vehicles.

Reliabilities of the subvertices ($subVrlb$) are computed as the weighted sum of the factors where the weights ($rlbW$) are decided according to their importance based on line completeness (19). Note that, for incomplete vertices, as the subvertex 3-D location changes fast (as the vehicle enters or exits or changes view), clustering divergence (σ') is bound to be high and need not be considered for reliability consideration. However, in this case, angular similarity is more important than that for the complete vertex case. This knowledge is reflected in $rlbW$ selection in (19). In case of complete vertices, factors for completion (C) and location similarity ($1/(1+D)$) are already resolved, and hence, fewer relative weights (importance) are assigned than the factors for angular similarity of the connected line ($1-\mathrm{E}$) and cluster congruity ($1/(1+\sigma')$), and we assign $rlbW = [1\ 1\ 2\ 5]/9$. For incomplete vertices, the cluster of corresponding subvertex locations is not reliable, yet the angular similarity of the line connected ($1-\mathrm{E}$) is more likely to be similar to that of generic or previous incremental 3-D model; hence, we select $[1\ 1\ 5]/7$.

2) Incremental Vertex Estimate: Incremental estimates V_{3D} are found by the weighted mean (21) of the corresponding visible subvertices (v_{3D}) with their reliabilities (19) as the weights. Vertex reliability ($Vrlb$) is the mean (22) of the $subVrlb$ values of the corresponding visible subvertices (v_{3D}) in the current frame.

3) Incremental Line Reliability: Line reliability scores are computed by (23) from the line disparity factors and the disparity and reliability of the terminal subvertices from (19). Reliability weights ($rlbW$) also change with the completeness of the line and the terminal vertices (more complete means more reliable). As disparity-based similarity ($1/(1+D)$) and subvertex reliability values ($subVrlb$) of incomplete vertices are expected to be low, they are given less weight [see (23)]. For example, when $V1_{\mathrm{3D}}$ is complete and $V2_{\mathrm{3D}}$ is incomplete, then ($1/(1+D_1)$) and $subVrl_1$ are expected to be high; ($1/(1+D_2)$) and $subVrlb_2$ are to be low; and the values of $subVrlb$ are more reliable than disparities. Hence, we have selected $rlbW = [2\ 4\ 2\ 1\ 4\ 2]/15$. When both $V1_{\mathrm{3D}}$ and $V2_{\mathrm{3D}}$ are complete, symmetric relative weights are used, with $rlbW = [1\ 4\ 2\ 2\ 4\ 4]/17$, where less weight is assigned to C as it is less relevant there.

4) Model Reliability: Once we have the 3-D vertex locations and their 2-D connectivity (from the $Pt2LnCorr$ matrix), we practically have the entire wire frame 3-D model incrementally built. Hence, for computing the unified model reliability ($Mrlb$) over the frame sequence, we only consider the reliabilities of the visible vertices in the current frame and use the mean as the incremental model reliability, as in (24).

D. Feature Extraction by Structural Tracking

Due to the smooth surface boundaries of automobiles, automatic 2-D corner detection in real video data is very error prone. Shadows, specular reflections, etc., in the uncontrolled traffic video add to the complexity. For this reason, we manually initialize the 2-D corners and connectivity between them (the *Pt2LnCorr* matrix) *only* in the frame where they appear for the first time, and subsequently, we automatically track, predict, and extract the corners and lines for all the other frames. Note that the surveillance video for a particular vehicle is often short. The *fully* automated feature extraction is not the focus of this paper. Existing methods, such as in [21], [24], [25], [27], and [39], can contribute toward the development of a fully automated system. Tracking of individual corners may change the rigid structure of a vehicle in 3-D, which is not expected. Thus, we use the entire *visible* vehicle structure to track features over the frames. The key steps are summarized in Algorithm 3.

Algorithm 3: Feature extraction by structural tracking.

1. Detect moving vehicle by image subtraction and morphological cleaning. Then, get the merged bounding box that is tracked across the frames.
2. Get image evidences of the 2-D corner and line for tracking and feature reliability of the predictions.
3. Manually initialize the 2-D corners and line connectivity for the first frame and for frames where new features are encountered.
4. Form a corner structure to pass across the frames.
5. Track the wire frame (formed by 2-D corners and connecting lines) in its entirety by cross-correlation-based neighborhood search.
6. Estimate 2-D corners for the current frame from the tracked wire frame.

1) Moving Vehicle Detection: In the preprocessing, we take out the even field of every frame and convert color ($\mathrm{R} =$ red, $\mathrm{G} =$ green, $\mathrm{B} =$ blue) frames to a grayscale one ($\mathrm{I} = (\mathrm{R}+\mathrm{G}+\mathrm{B})/3$) to reduce the computational complexity. The first frame of the video sequence is used as the base frame that is subtracted from every following frame to get the moving regions in that frame. To extract silhouettes of the moving regions, at first, we do binarization, and then, several morphological operations are employed, five times each, in this order: 1) spurious noise (isolated dangling edges) removal; 2) morphological cleaning; 3) morphological closing; and 4) majority filtering. Then, the connected components of the moving regions are labeled with 8-neighorhood connectivity. Regions with an area of less than 10 pixels are discarded. A minimum rectangular bounding box of the moving region is

TABLE IV
CORNER STRUCTURE: EACH LOCATION IN [COL ROW] FORMAT

Field	Dim.	Description
GenNum	[1x1]	Stores corner number as per the 3D generic vehicle model
Loc	[1x4]	[1x2]: 2D corner location in current frame; [1x2]: Bounding Box (BB) position in entire frame
prevLoc	[1x4]	[1x2]: 2D corner location in the previous frame; [1x2]: BB position in the previous entire frame
M_HT	[NxN]	2D hash table of the motion vectors so far seen of the current corner over the frame sequence, where N is flexible based on how many unique vectors are encountered
Motion	[1x2]	2D motion vector of corner by votes in the M_HT
PredCntr	[1x4]	[1x2]: 2D centroid of motion based predictions; [1x2] BB
CrnRlb	[1x1]	Corner reliability (overall) based on the frame evidences
LineCnt	[1x1]	Number of lines concurrent at the corner
Line	[1xLineCnt]	Structure for the lines concurrent at the corner: *Next 3 rows define the structure of each Line*
OtherCrn	[1x4]	[1x2]: 2D location of other corner of line; [1x2]: BB
Slope	[1x1]	2D slope angle of the line in [X_{img}-Y_{img}] format
LnRlb	[1x1]	Reliability of the detected line based on the frame evidences

tracked over the frames to reduce the computational complexity of searching the particular vehicle.

2) Evidences for Tracking and Feature Reliability: We use cross-correlation-based neighborhood matching for tracking and finding the confidence score of the automatically extracted features. In addition to intensity information, we also find the edge directions $(carEdir)$ and magnitudes $(carEmag)$ for the cropped areas. We first smooth by a 3×3 Gaussian filter with standard deviation (*sigma*) equal to 0.5. Then, we use second-order discrete partial derivatives to find the direction and magnitude of the edges. We consider directional angles only in 30° of intervals. We extract the silhouette $(carSilh)$ and boundary ($carBnd$, which is computed from the perimeter of $carSilh$) of the moving vehicle. This is extracted, because, due to the smoothness of the surfaces, edges are not always definitive, and $carBnd$ can help to decide the presence of lines between two corners. Thus, for each frame, we get an evidence set *Evd* (gray image, edge direction, edge magnitude, moving silhouette, and moving boundary). This is used for finding the corners and lines of the moving vehicle in the video.

3) Corner Structure for Passing Information Over Frames: We use a data structure (see Table IV) for storing frame-based information on the corners and the lines connecting them, and we pass the structure to the next frame to predict the new 2-D locations of the corners and lines.

4) Wire-Frame Global Tracking: Local motion-based tracking and prediction of individual corners may become zigzag due to noisy cross correlations and 2-D motion vectors for different 2-D corners may completely be different. This may severely deform the wire-frame structure, which is unexpected for rigid objects such as vehicles. A solution is to use the structural constraints between the 2-D corners. However, it is hard to decide the starting point (with true prediction) for applying structural constraints. The entire wire frame can be changed by small local deformations to fit the 2-D features extracted without breaking the structural interrelations among them.

We reduce the aforementioned complexity by globally tracking the entire wire frame (comprising 2-D corners and 2-D lines connecting them). We predict the position of the entire wire frame by the best *global* cross correlation from the previous frame to the current frame. The current 2-D features are estimated from this predicted wire frame. When the view of the moving vehicle severely changes and currently tracked wire frame cannot fit the new view of the vehicle, relevant corners are re-initialized. Thus, we consider geometrical deformation of the frame. Note that we have considered global cross correlation in this paper. The 3-D model estimated in this paper can be used for obtaining the best global transformation of the vehicle.

For tracking the wire frame, different evidences from the previous and current frames are used. A [7×7] window around the previous location of each corner is searched, and the pixel with the highest cross correlation is selected as the current location. Cross correlations are measured for both frames over 5×5 pixels patches for evidences. The entire wire frame (with only complete corners) is moved by exactly the same amount so that each of the moved corners is within the search window for its previous frame counterpart. By

$$confFr = \sum_{crn:\text{all visible corners}} conf(crn). \tag{25}$$

we sum up the confidence scores $[conf\ (crn)]$ of all complete corners (crn) to get the single confidence value $(confFr)$ of the entire wire frame for a position. Based on the highest confidence $(confFr)$ of the entire wire frame, we select its predicted position.

Individual $[conf\ (crn)]$ is computed by linear fusion of similarities between the corner (crn) in the previous frame and the choices in the current frame (within the 7×7 neighborhood window) by

$$\begin{aligned} conf(crn) = 100 * \ (&\alpha_1 * GrayCorr + \alpha_2 * \ EmagCorr \\ &+ \alpha_3 * \ EdirSim + \alpha_4 * \ proxIndx) \\ &\text{where } \sum_i \alpha_i = 1. \end{aligned} \tag{26}$$

The values of α_i's were determined as $\alpha_1 = 0.6$, $\alpha_2 = 0.2$, $\alpha_3 = 0.1$, and $\alpha_4 = 0.1$ based on experiments. The similarity measures [in (26)] are computed as follows:

a) *Grayscale value-based cross-correlation index (GrayCorr):* This measures intensity-based similarity between 5×5 patches around the pixel locations in the previous and current frames, as in (27), shown at the bottom of the next page. $Xcorr(x, y)$ is the correlation index between two intensity image patches $[x]_{N\times N}$ and $[y]_{N\times N}$ with $N = 5$. Individual elements $[R_{xy}\ (m, n)]$ of the correlation matrix are computed, as shown in (27). When x and y are the same image patches, we get an autocorrelation matrix [like $GrayAcorr$ in (27) and $EmagAcorr$ in (28), shown at the bottom of the next page], and when x and y are different, we get the cross-correlation matrix [like $GrayXcorr$ in (27) and $EmagXcorr$ in (28)]. Only the central elements $(a1, b1)$ are used to compute the single similarity index, like grayscale correlation index $(GrayCorr)$ in (27).

b) *Edge-magnitude-based cross correlation* $(EmagCorr)$*:* This measures the similarity of edge-magnitudes for the 5×5 patches around the pixels by (28) [using (27)], where $EmagXcorr$ and $EmagAcorr$ are cross- and autocorrelation matrices, and a2 and b2 are the respective center elements.

c) *Edge-direction similarity* $(EdirSim)$*:* This measures the similarity of the edge directions by

$$EdirSim = \cos\left(|currEdir[i,j] - prevEdir[m,\ n]|\right) \quad (29)$$

i.e., $prevEdir$ and $currEdir$, at pixel (m,n) in the last frame and pixel (i,j) in the current frame, respectively.

d) *Proximity index* $(proxIndx)$*:* This measures distances between the previous frame corner location (m,n) and the candidate pixel (i,j) in the current frame, i.e.,

$$proxIndx = [1/[1 + \|(i,j) - (m,n)\|]]. \quad (30)$$

5) Estimation of 2-D Corners for the Current Frame: From the tracking result of the entire wire frame, we get the global motion, which, in our case, is also the local motion. We apply this motion to each of the complete corners in the previous frame to get the estimated locations in the current frame (see Fig. 4). Now, the 2-D linear edge segments can be updated based on corner connectivity information from the previous corner structure. Incomplete corners are those not yet seen, and the connecting lines to them are not complete. These incomplete corners are predicted by extending the corresponding line in the previous frame, until the line hits the boundary of the current frame (generally the left or top margin). Note that we have used a search window (7×7) that is large enough such that corresponding corners are safely within the window. In addition, due to the frame-subtraction-based silhouette detection of moving vehicles (described earlier), the search window never falls back, and it is always current. Thus, we have not considered the use of an adaptive window whose size may vary with the speed of the motion.

The line reliability values are updated for the current frame by counting the number of nonzero pixels (nnz) in the close neighborhood of line L_n, which was normalized by the length of the line $(L_{2\mathrm{D}})$, as in

$$Erlb(L_n) = 100 * \min\left[1, \max\left[\frac{nnz(Emag)}{1+L_{2\mathrm{D}}}\ \frac{nnz(silhBnd)}{1+L_{2\mathrm{D}}}\right]\right]. \quad (31)$$

Edge magnitude $(Emag)$ is not always a good evidence for line reliability because of the blunt ridges of the vehicles. Hence, we use both $Emag$ and silhouette boundary $(silhBnd)$ to compute reliabilities as evidences, and the maximum of them is selected.

Finally, the corner reliabilities are computed by the mean of the reliabilities of the concurring lines, i.e.,

$$CrnRlb = \frac{1}{LineCnt} \sum_{i=1}^{LineCnt} Erlb(i). \quad (32)$$

When any new corner or line feature appears in the video sequence, it is initialized for the first time and then automatically tracked with the other parts of the wire frame in the following frames.

$$\begin{aligned}
Xcorr(x,y) &= \lfloor R_{xy} \rfloor_{(2N-1)\times(2N-1)}, \text{ where } [x]_{N\times N} \text{ and } [y]_{N\times N}, \quad \text{and} \quad N = 5\\
R_{xy}(m,n) &= \begin{cases} \sum_{i=0}^{N-m-1} \sum_{j=0}^{N-n-1} x[i+m, j+n] * y[i,j], & m,n \geq 0 \\ R_{xy}(-m,-n), & m,n < 0 \end{cases} \text{ where } m,n \in [-N,N]\\
GrayXcorr &= Xcorr(currGrayPatch,\ prevGrayPatch)\\
GrayAcorr &= Xcorr(currGrayPatch,\ currGrayPatch)\\
a1 &= GrayXcorr[N,N] : \text{Center element of the square matrix}\\
b1 &= GrayAcorr[N,N] : \text{Center element of the square matrix}\\
GrayCorr &= [1/[1 + (|a1 - b1|/(a1 + b1))]]
\end{aligned} \quad (27)$$

$$\begin{aligned}
EmagXcorr &= Xcorr(currEmagPatch,\ prevEmagPatch)\\
EmagAcorr &= Xcorr(currEmagPatch,\ currEmagPatch)\\
a2 &= EmagXcorr\ [N,N] : \text{Center element of the square matrix } and\ N = 5\\
b2 &= EmagAcorr[N,\ N] : \text{Center element of the square matrix } and\ N = 5\\
EmagCorr &= [1/[1 + (|a2 - b2|/(a2 + b2))]]
\end{aligned} \quad (28)$$

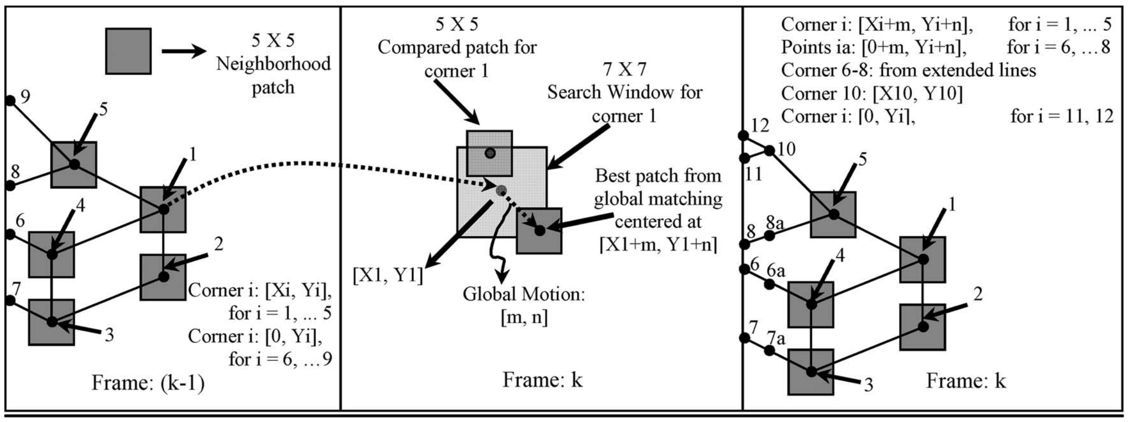
5 X 5 Neighborhood patch
Corner i: [Xi, Yi], for i = 1, ... 5
Corner i: [0, Yi], for i = 6, ...9
Frame: (k-1)
5 X 5 Compared patch for corner 1
7 X 7 Search Window for corner 1
Best patch from global matching centered at [X1+m, Y1+n]
[X1, Y1]
Global Motion: [m, n]
Frame: k
Corner i: [Xi+m, Yi+n], for i = 1, ... 5
Points ia: [0+m, Yi+n], for i = 6, ...8
Corner 6-8: from extended lines
Corner 10: [X10, Y10]
Corner i: [0, Yi], for i = 11, 12
Frame: k

Fig. 4. Feature-extraction method. Frame $(k-1)$: Neighborhood patches of 5×5 pixels are considered around each complete corner (1–5). Frame k: A 7×7 search window is used to move around the 5×5 patch for best match. Best match is found based on the total reliability of all the complete corners. Global motion $[m, n]$ is used to predict complete corners (1–5) in frame k. New corner 10 is initialized for frame k and tracked thereafter. Incomplete corners (6–8, 11, and 12) are found by extending corresponding linear edge segments until they cut the left margin of the frame.

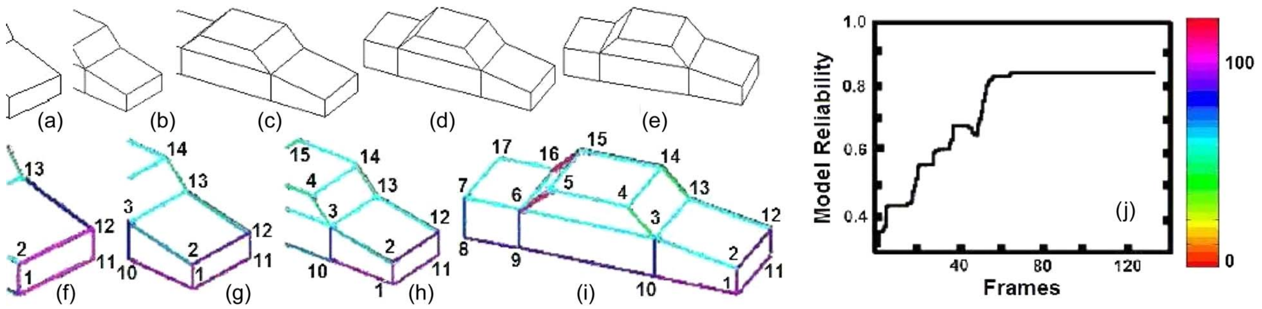

Fig. 5. Simulated sedan. (a)–(e) Sample frames (total number of frames: 125). (f)–(i) Incremental models at different stages. Reliabilities of lines are color coded. (j) Unified model reliability value of the incremental 3-D model over the frame sequence (best viewed in color).

E. Summary of the Proposed Method

As summarized in Fig. 1, we start with a traffic video from a single static uncalibrated camera where, often, the vehicles are partially seen. After motion-based vehicle detection and initialization of the 2-D corner and lines in the frame the first time they are encountered, 2-D features in subsequent frames are automatically extracted (see Algorithm 3 in Section III-D) using the local cross-correlation-based global wire-frame track-predict-verify method (see Fig. 4). The detected 2-D features are mapped to 3-D (see Figs. 2 and 3) using a novel directional template library (DTL) approach (see Algorithm 1) for each video frame. Three-dimensional vertices and ridges form the single-frame-based 3-D model (see Section III-B1). A novel adaptive-clustering approach (see Algorithm 2) incrementally adds and updates the 3-D model parameters as increasingly more views of the vehicle are partially or completely seen (see Section III-B2) using an exponential forgetting and weighted history. The incrementally learned 3-D model until the last frame is used as the ground-truth 3-D model at the current frame. Reliability scores are then computed to measure the structural congruity of the 3-D model with this ground truth at different abstraction levels (see Section III-C). These reliability scores also act as the weight factors during incremental view integration (see Sections III-B2d–B2i) and provide the final incremental 3-D model at the current time point (video frame).

IV. Experimental Results and Discussions

A. Results on Simulated Traffic Data

1) Simulated Traffic Data: A ten-vertex 15-surface block-based vehicle was generated, and its motion was simulated with both translation and orientation changes over the frames. Sample frames are shown in Fig. 5(a)–(e). In (20), we have selected a scale factor of exponential forgetting Q to be 0.5.

2) Results on Simulated Data: Incremental results for frames 17, 35, 50, and 100 are shown in Fig. 5(f)–(i). The numbers of the vertices are shown. Edge segments are color coded according to reliability values from red (reliability 0) to violet (reliability 1). The overall reliability of the model is 0.85. The model reliability value over the complete video sequence (125 frames) is shown in Fig. 5(j). As expected, the reliability value gradually increases (see Fig. 6) as more frames are seen with minor deviations due to newly seen vertices affecting other

Generic Vertex #	Ground Truth	Single-frame based 3D location estimates		Clustering based 3D location estimates & model reliabilities			
		Frame: 50	Frame: 100	Frame: 50	Rlb in Fr: 50	Frame: 100	Rlb in Fr: 100
1	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	1.000	[0, 0, 0]	1.000
2	[0, 0, 10]	[0, 0, 10]	[0, 0, 10]	[0, 0, 10]	0.836	[0, 0, 10]	0.865
3	[0, 30, 20]	[0, 30, 16]	[0, 30, 16]	[1, 28, 15]	0.408	[1, 30, 16]	0.449
4	[0, 40, 40]	[0, 42, 27]	[0, 42, 27]	[4, 39, 26]	0.267	[2, 41, 26]	0.344
5	[0, 70, 40]	Not Seen	[0, 72, 27]	Not Seen	Not Seen	[2, 68, 26]	0.330
6	[0, 80, 20]	Not Seen	[0, 81, 16]	Not Seen	Not Seen	[0, 77, 16]	0.404
7	[0, 100, 20]	Not Seen	[0, 101, 16]	Not Seen	Not Seen	[2, 94, 16]	0.386
8	[0, 100, 0]	Not Seen	[0, 100, 0]	Not Seen	Not Seen	[0, 95, 0]	0.591
9	[0, 80, 0]	Not Seen	[0, 80, 0]	Not Seen	Not Seen	[0, 74, 0]	0.611
10	[0, 30, 0]	[0, 29, 0]	[0, 30, 0]	[0, 26, 0]	0.651	[0, 28, 0]	0.717
11	[30, 0, 0]	[30, 0, 0]	[30, 0, 0]	[29, 0, 10]	0.715	[30, 0, 0]	0.765
12	[30, 0, 10]	[30, 0, 10]	[30, 0, 10]	[29, 0, 16]	0.686	[30, 0, 10]	0.731
13	[30, 30, 20]	[30, 30, 16]	[30, 30, 16]	[29, 31, 16]	0.399	[30, 30, 16]	0.425
14	[30, 40, 40]	[30, 42, 27]	[30, 42, 27]	[29, 42, 27]	0.335	[30, 42, 27]	0.371
15	[30, 70, 40]	[30, 70, 27]	[30, 73, 27]	[28, 59, 26]	0.272	[29, 69, 27]	0.345
16	[30, 80, 20]	Not Seen	[30, 62, 21]	Not Seen	Not Seen	[26, 61, 23]	0.370
17	[30, 100, 20]	Not Seen	[30, 82, 21]	Not Seen	Not Seen	[28, 78, 22]	0.344
18	[30, 100, 0]	Not Seen	Not Seen	Not Seen	Not Seen	Not Seen	Not Seen
19	[30, 80, 0]	Not Seen	Not Seen	Not Seen	Not Seen	Not Seen	Not Seen
20	[30, 30, 0]	Not Seen	Not Seen	Not Seen	Not Seen	Not Seen	Not Seen

Fig. 6. Simulated sedan: The ground-truth 3-D locations of the vertices in simulated data and their estimated locations at frames 50 and 100 are shown in the table. Clustering-based 3-D estimates are robust than their single-frame-based counterparts. Generally, the reliabilities of the 3-D vertices increase when more frames are considered.

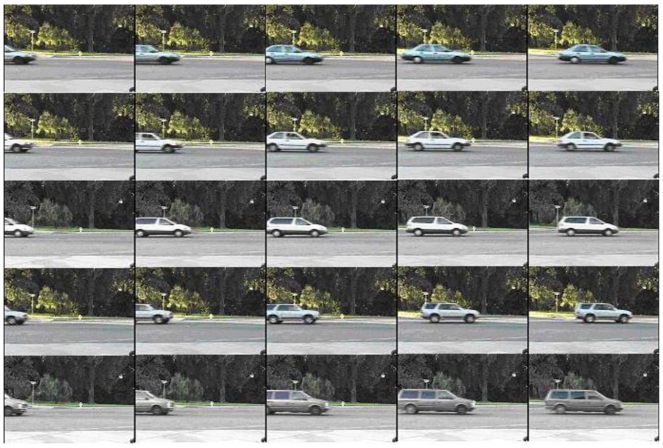
Fig. 7. Sample frames of vehicular videos taken at traffic spot 1 (*best viewed in color*).

estimations. Note that vertices 18, 19, and 20 are never seen over the entire video sequence.

B. Results for Real Traffic Video Data

For unsupervised learning, without any ground truth, for traffic vehicles in real video, the incremental model learned until the previous video frame is used as the ground truth, and corresponding parameters and reliabilities are used in the computation of the values for the current frame.

1) Real Traffic Video Data: Real traffic video data have been collected by an uncalibrated camera in two right-angle street curves so that many different views of the vehicle are seen. Sample frames for the selected vehicles are shown in Figs. 7 and 8. For traffic spot 1 (see Fig. 7), vehicles are at a relatively closer distance, and there are fewer views and video frames for each vehicle, whereas, in traffic spot 2 (see Fig. 8), we get a larger number of views and video frames. This is expected to improve the 3-D model-building results. Note that, in the short duration (60–383 frames: 2–13 s) of the traffic videos used in this paper, we do not expect significant illumination changes. The small illumination changes are automatically handled by wire-frame tracking (see Section III-D) by best-match-based correspondences across consecutive frames. However, for longer videos, where illumination changes will be important, an approach such as [25] can be used. In addition, as one video is independent of the others, illumination differences between two videos will not affect the results. Furthermore, there are no significant motion outliers from shadow [52].

2) Parameters: For all traffic video sequences, we have considered vehicles that a) enter through the left side margin of the frame, b) move toward the right side margin, and then c) either exit through the right side margin of the frame (traffic spot 1; see Fig. 7) or turn right toward the camera and exit through the bottom side margin of the frame (traffic spot 2; see Fig. 8). We use angular similarity tolerance (*angSimTh*) to enforce parallelism and to decide similarity between lines in the models at different incremental stages. Note that most of the traffic vehicles are symmetric and indeed have parallel 3-D ridges that are mapped to 2-D parallel edges. The threshold *angSimTh* can be learned from several traffic videos. For traffic videos taken at spot 2, due to small lengths of the lines, one pixel error in the 2-D corner location may magnify the error in angular the directions; hence, we have

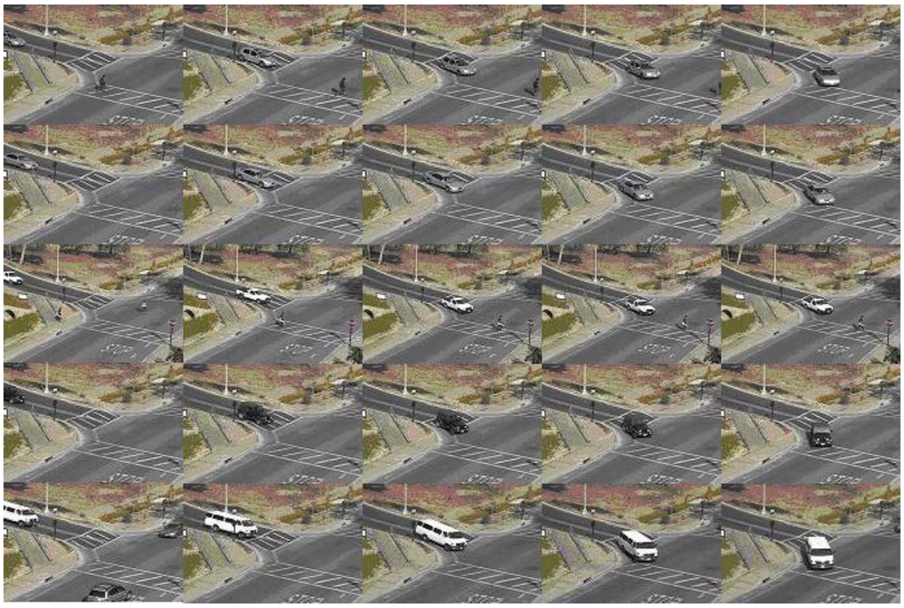

Fig. 8. Sample frames of vehicular videos taken at traffic spot 2 (*best viewed in color*).

used $angSimTh = 10°$. On the other hand, for traffic videos taken at spot 1, due to larger lengths of the lines detected, the estimation error in the 2-D corner location causes much less error in the angular direction; hence, $angSimTh = 5°$ is used. As the distance between the vehicle and the camera is not known, single-frame-based estimates of the 3-D vertex locations are valid up to a scale factor. We normalize the coordinates to make the width of the vehicle to be 30 units in OCC. This scale remains the same for nearly all vehicles due to fixed lane width in standard highways. Note that the selected width of 30 units is arbitrary just to normalize all types of vehicles to compare their 3-D models, and any other fixed number could have been used as well. For real traffic data, vehicle view changes are fast. In exponential forgetting in (20) for subvertices $(v_{3\text{D}})$ and their reliability values $(subVrlb)$, we use exponential forgetting factor $Q = 0.7$. For videos from the same traffic spot, all the parameters are held constant.

3) Experimental Results on Real Traffic Data: The results for Car1, SUV2, Pickup5, and Jeep3 are shown in Figs. 9–12, respectively. In each of the figures, we show sample frames with superimposed features, incremental models learned until that time, with individual reliability values color coded, and overall learning curve for the estimated 3-D models. For the rest of the vehicle videos, we show the final 3-D model learning curves in Fig. 13. Compared with our earlier work [53], this paper provides the details of DTL, incremental learning and feature extraction, and results on several types of vehicles from different camera locations.

4) Discussion of Results: For vehicles in traffic spot 1 (see Figs. 9 and 10), estimations of vertices in the distant side surface are not robust due to near fronto-parallel views of the vehicles. On the other hand, for vehicles in traffic spot 2

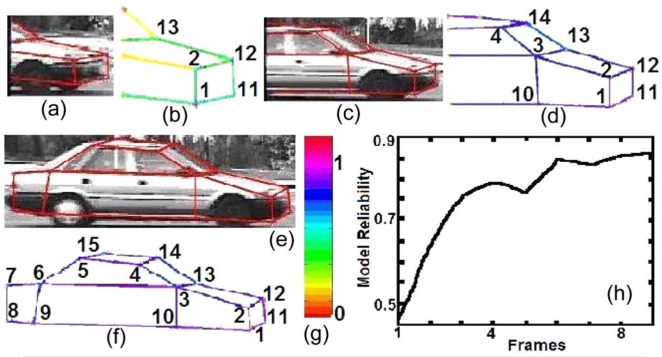

Fig (a-b) Frame: 9 3D Orientation: 78°		Fig (c-d) Frame: 16 3D Orientation: 74°	
1: [0, 0, 0] (1.00)	11: [30, 0, 0] (0.67)	1: [0, 0, 0] (1.00)	10: [0, 77, 0] (0.92)
2: [0, 0, 17] (0.62)	12: [30, 0, 17] (0.95)	2: [0, 0, 24] (0.92)	11: [30, 0, 0] (0.67)
	13: [30, 54, 17] (0.49)	3: [0, 78, 31] (0.88)	12: [30, 0, 22] (0.65)
		4: [0, 119, 55] (0.87)	13: [30, 67, 26] (0.65)
			14: [30, 113, 52] (0.88)

Fig (e-f)	Frame: 23		3D Orientation: 72°
1: [0, 0, 0] (1.00)	5: [0, 231, 62] (0.90)	9: [0, 278, 0] (0.91)	13: [29, 83, 26] (0.66)
2: [0, 0, 26] (0.93)	6: [0, 275, 32] (0.72)	10: [0, 92, 0] (0.93)	14: [30, 139, 54] (0.89)
3: [0, 92, 35] (0.90)	7: [0, 316, 33] (0.59)	11: [30, 0, 0] (0.67)	15: [26, 212, 54] (0.61)
4: [0, 140, 59] (0.88)	8: [0, 316, 0] (0.60)	12: [30, 0, 26] (0.95)	

Fig. 9. Traffic spot 1: Car1. (a), (c), and (e) Feature superimposed sample video frames. (b), (d), and (f) Incremental 3-D model until frames in (a), (c), and (e), respectively. Lines are color coded by their reliability values. (g) Color-coding bar. (h) Learning curve of the incremental 3-D model. The table shows the estimated locations (inside square brackets) and reliabilities (inside parentheses) of the vertices (*best viewed in color*).

(see Figs. 11 and 12), as more views are visible in the video, vertices on the distant side surfaces are robustly estimated. For traffic spot 2, as view variation is nearer the end of video sequences, the wire-frame tracking and feature estimations are not robust; hence, the model learning trends start to drop down [see Figs. 11(h), 12(h), and 13(d)–(f)]. For traffic spot 1 with less view changes, this trend is not seen [see Figs. 9(h), 10(h), and 13(a)–(c)].

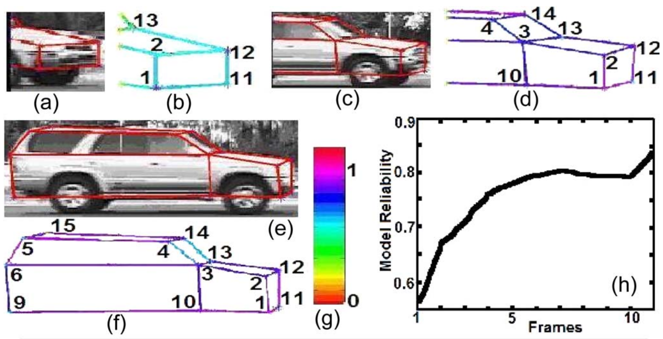

Fig (a-b) Frame: 7	3D Orientation: 83°	Fig (c-d) Frame: 15	3D Orientation: 78°
1: [0, 0, 0] (0.97)	11: [30, 0, 0] (0.67)	1: [0, 0, 0] (1.00)	10: [0, 71, 0] (0.91)
2: [0, 0, 15] (0.62)	12: [30, 0, 15] (0.96)	2: [0, 0, 30] (0.91)	11: [30, 0, 0] (0.67)
	13: [30, 42, 15] (0.52)	3: [0, 71, 32] (0.87)	12: [30, 0, 32] (0.93)
		4: [0, 103, 53] (0.87)	13: [30, 62, 32] (0.70)
			14: [30, 98, 54] (0.88)
Fig (e-f)		Frame: 25	3D Orientation: 72°
1: [0, 0, 0] (1.00)	5: [0, 436, 110] (0.65)	11: [30, 0, 0] (0.67)	15: [29, 360, 94] (0.32)
2: [0, 0, 56] (0.91)	6: [0, 500, 68] (0.69)	12: [30, 0, 57] (0.93)	
3: [0, 110, 53] (0.89)	9: [0, 473, 0] (0.59)	13: [30, 93, 45] (0.69)	
4: [0, 110, 85] (0.87)	10: [0, 116, 0] (0.93)	14: [30, 108, 84] (0.87)	

Fig. 10. Traffic spot 1: SUV2. (a), (c), and (e) Feature superimposed sample video frames. (b), (d), and (f) Incremental 3-D model until frames in (a), (c), and (e), respectively. Lines are color coded by their reliability values. (g) Color-coding bar. (h) Learning curve of the incremental 3-D model. The table shows the estimated locations (inside square brackets) and reliabilities (inside parentheses) of the vertices (*best viewed in color*).

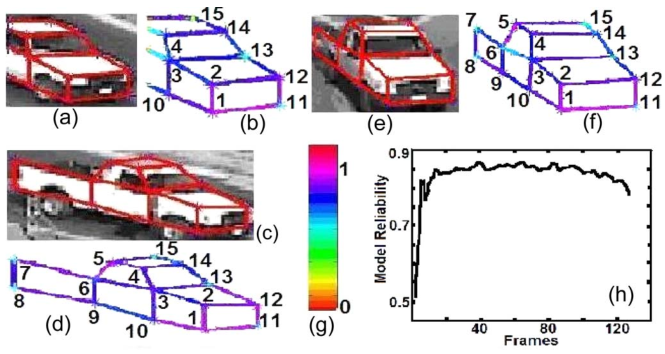

Fig (a-b) Frame: 21	3D Orientation: 90°	Fig (c-d) Frame: 80	3D Orientation: 87°
1: [0, 0, 0] (0.99)	11: [30, 0, 0] (0.67)	1: [0, 0, 0] (1.00)	9: [0, 72, 5] (0.85)
2: [0, 0, 14] (0.92)	12: [30, 0, 14] (0.94)	2: [0, 0, 17] (0.93)	10: [0, 33, 0] (0.87)
3: [0, 24, 20] (0.86)	13: [30, 24, 19] (0.53)	3: [0, 33, 23] (0.91)	11: [30, 0, 0] (0.67)
4: [1, 29, 31] (0.83)	14: [30, 34, 28] (0.90)	4: [0, 42, 38] (0.90)	12: [30, 0, 14] (0.97)
10: [0, 24, 0] (0.91)	15: [30, 55, 28] (0.30)	5: [0, 62, 38] (0.94)	13: [30, 33, 23] (0.55)
		6: [0, 72, 23] (0.75)	14: [30, 51, 36] (0.82)
		7: [0, 126, 23] (0.62)	15: [30, 71, 39] (0.54)
		8: [0, 126, 6] (0.62)	
Fig (e-f)		Frame: 219	3D Orientation: 98°
1: [0, 0, 0] (0.98)	5: [0, 31, 30] (0.92)	9: [0, 38, 0] (0.88)	13: [30, 17, 17] (0.51)
2: [0, 0, 12] (0.92)	6: [0, 38, 15] (0.64)	10: [0, 20, 0] (0.88)	14: [30, 30, 25] (0.79)
3: [0, 20, 18] (0.83)	7: [0, 56, 17] (0.54)	11: [30, 0, 0] (0.67)	15: [29, 39, 28] (0.52)
4: [0, 20, 30] (0.87)	8: [0, 56, 0] (0.61)	12: [30, 0, 12] (0.93)	

Fig. 11. Traffic spot 2: Pickup5. (a), (c), and (e) Feature superimposed sample video frames. (b), (d), and (f) Incremental 3-D model until frames in (a), (c), and (e), respectively. Lines are color coded by their reliability values. (g) Color-coding bar. (h) Learning curve of the incremental 3-D model. The table shows the estimated locations (inside square brackets) and reliabilities (inside parentheses) of the vertices (*best viewed in color*).

Contrary to the expectation, the model reliability is not monotonically increasing. This is due to the appearance of new edge segments and vertices that are sometimes noisy and affect previous estimates and, hence, the reliability values. Note that wire-frame tracking could handle small occlusion due to the "stop sign" in video at traffic spot 2 as unoccluded features correctly tracked the occluded features.

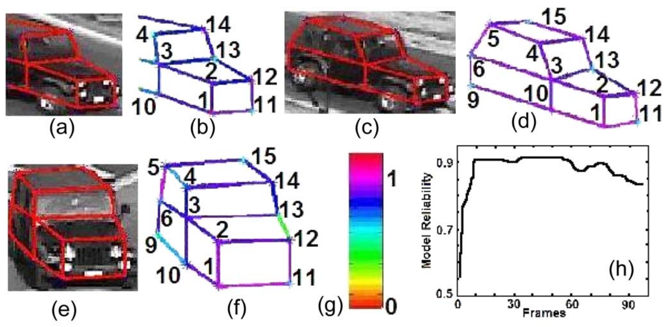

Fig (a-b) Frame: 20	3D Orientation: 89°	Fig (c-d) Frame: 70	3D Orientation: 87°
1: [0, 0, 0] (1.00)	11: [30, 0, 0] (0.67)	1: [0, 0, 0] (1.00)	9: [0, 134, 0] (0.64)
2: [0, 0, 20] (0.92)	12: [30, 0, 21] (0.94)	2: [0, 0, 21] (0.96)	10: [0, 53, 0] (0.96)
3: [0, 46, 21] (0.90)	13: [30, 44, 23] (0.55)	3: [0, 53, 21] (0.96)	11: [30, 0, 0] (0.67)
4: [1, 53, 40] (0.62)	14: [30, 47, 47] (0.96)	4: [0, 65, 46] (0.95)	12: [30, 0, 20] (0.96)
10: [0, 44, 0] (0.91)		5: [0, 116, 47] (0.97)	13: [30, 53, 21] (0.56)
		6: [0, 134, 21] (0.73)	14: [30, 59, 50] (0.95)
			15: [30, 114, 50] (0.64)
Fig (e-f)		Frame: 125	3D Orientation: 98°
1: [0, 0, 0] (0.98)	4: [0, 23, 34] (0.81)	9: [0, 43, 6] (0.50)	12: [30, 0, 19] (0.88)
2: [0, 0, 19] (0.94)	5: [0, 38, 34] (0.85)	10: [0, 22, 0] (0.82)	13: [30, 22, 20] (0.49)
3: [0, 22, 20] (0.89)	6: [0, 43, 20] (0.67)	11: [30, 0, 0] (0.67)	14: [30, 22, 34] (0.87)
			15: [30, 38, 34] (0.59)

Fig. 12. Traffic spot 2: Jeep3. (a), (c), and (e) Feature superimposed sample video frames. (b), (d), and (f) Incremental 3-D model until frames in (a), (c), and (e), respectively. Lines are color coded by their reliability values. (g) Color-coding bar. (h) Learning curve of the incremental 3-D model. The table shows the estimated locations (inside square brackets) and reliabilities (inside parentheses) of the vertices (*best viewed in color*).

V. Conclusion

This paper has described a learning-based incremental 3-D modeling approach for vehicles from traffic videos captured by a single static uncalibrated camera. The DTL approach has effectively overcame the foreshortening effect in perspective projection and mapped 2-D features to 3-D. Unsupervised learning has utilized the incrementally learned 3-D model itself to gradually improve the estimations of features and the model as more video frames are considered. Note that, by unsupervised learning, we mean that we do not need a CAD model of the vehicle or any kind of ground-truth 3-D model. The incremental model estimated until the last frame is used instead (with exponential forgetting and weighted history). The methodology considered 3-D model building of rigid objects where a single generic 3-D model is utilized. As the method worked quite well with uncalibrated and noisy data, it has the potential in applications with streaming video information. Although the focus in this paper has been 3-D model reconstruction of vehicles from video, we have used a wire-frame tracking method to extract 2-D vehicle features. In the future, we plan to develop a sophisticated method for automated vehicle detection [54] and tracking [44]–[49], [52] to improve our incremental 3-D reconstruction results. The 3-D models developed in this paper will help other ITSs, such as freeway merging [55], urban traffic monitoring [56], different tolling amount in electronic toll booths [57], and providing automated parking information [58] to a particular entering vehicle. In future extensions to a wider variety of vehicles, occlusion from multiple vehicles (similar to [38]), the effect of reflection and illumination variations (like in [25]), and adaptation of the exponential forgetting parameter from the video data itself will be considered. We also plan to integrate shadow rejection [52] algorithms to make the proposed method robust to such moving outliers in future work.

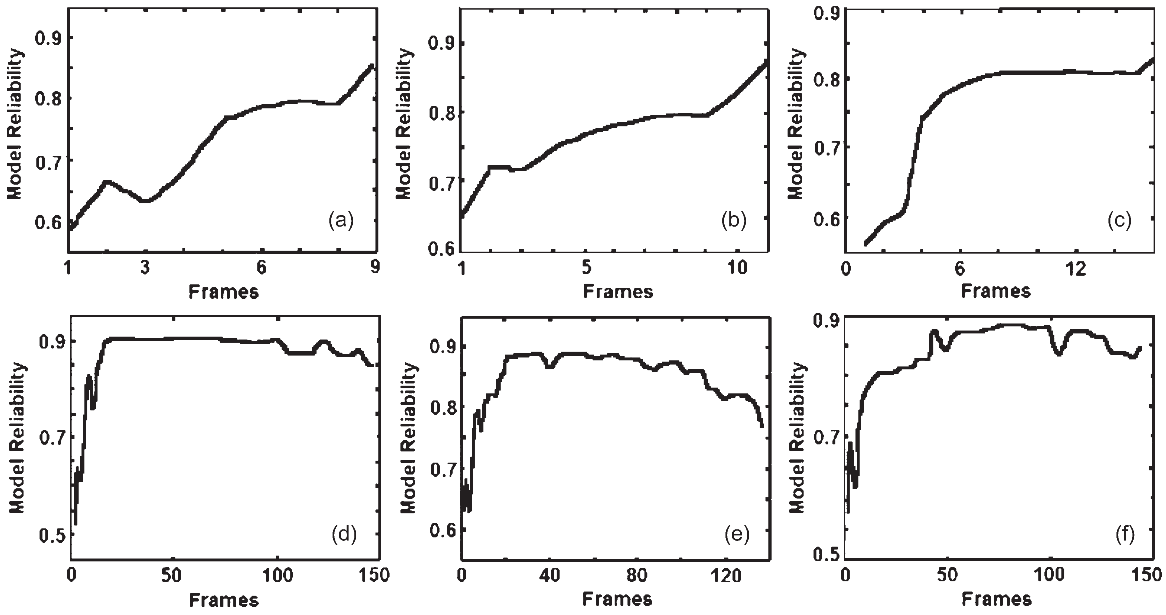

Fig. 13. Trends of unsupervised learning for other vehicles. Increasing reliability of the estimated incremental 3-D models of traffic spot 1. (a) Car 2. (b) SUV 1. (c) Van 1, traffic spot 2. (d) Car 12. (e) Car 14. (f) Van 3. Note that for the videos in traffic spot 1, due to fewer views of the vehicles (at closer distance) and fewer video frames per vehicle, corners in the far Y–Z plane (in OCC) are not visible, and the learning trend could not reach the steady state within the lengths of the videos. On the other hand, for the videos in traffic spot 2, near the ends of the videos, due to rapid view changes of the turning vehicles, wire-frame tracking estimates and predicted 3-D vertex locations are less reliable, which causes oscillations and drags down the overall 3-D model reliabilities. Better camera view angle and tracking methods are expected to improve the results (see text).

References

[1] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequence: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 377–391, Sep. 2008.

[2] R. Isukapalli and R. Greiner, "Efficient car recognition policies," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, pp. 2134–2139.

[3] G. L. Foresti, V. Murino, and C. Regazzoni, "Vehicle recognition and tracking from road image sequences," *IEEE Trans. Veh. Technol.*, vol. 48, no. 1, pp. 301–318, Jan. 1999.

[4] M. Betke, E. Haritaoglu, and L. S. Davis, "Highway scene analysis in hard real-time," in *Proc. IEEE Conf. ITS*, 1997, pp. 812–817.

[5] S. Nadimi, "Physics-based evolutionary strategies and dynamic sensor fusion for moving object detection," Ph.D. dissertation, Dept. Comput. Sci., Univ. Calif. Riverside, Riverside, CA, Jun. 2003.

[6] M. Kamachi, Y. Wu, and S. Ogata, "A vehicle recognition method robust against vehicles' overlapping based on stereo vision," in *Proc. IEEE Conf. ITS*, 1999, pp. 865–869.

[7] M. Kagesawa, S. Ueno, K. Ikeuchi, and H. Kashiwagi, "Local-feature based vehicle recognition in infra-red images using parallel vision board," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 1999, pp. 1828–1833.

[8] M. Kagesawa, S. Ueno, K. Ikeuchi, and H. Kashiwagi, "Recognizing vehicles in infrared images using IMAP parallel vision board," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 1, pp. 10–17, Mar. 2001.

[9] T. Ito, K. Yamada, and K. Nishioka, "Preceding vehicle recognition algorithm using fusion of laser radar and image processing," in *Proc. Intell. Vehicles Symp.*, 1993, pp. 420–425.

[10] T. R. Lim and A. T. Guntoro, "Car recognition using Gabor filter feature extraction," in *Proc. Asia-Pacific Conf. Circuits Syst.*, 2002, vol. 2, pp. 451–455.

[11] J. B. Kim, C. W. Lee, K. M. Lee, T. S. Yun, and H. J. Kim, "Wavelet-based vehicle tracking for automatic traffic surveillance," in *Proc. IEEE Reg. 10 Conf. Elect. Electron. Technol.*, 2001, vol. 1, pp. 313–316.

[12] S.-Y. Park and M. Subbarao, "Pose estimation and integration for complete 3D model reconstruction," in *Proc. IEEE WACV*, 2002, pp. 143–147.

[13] F. De Felice, T. Gramegna, F. Renna, G. Attolico, and A. Distante, "A portable system to build 3D models of cultural heritage and to allow their exploration by blind people," in *Proc. IEEE Int. Workshop Haptic Audio Visual Environ. Appl.*, 2005, pp. 1–6.

[14] J. Wu and X. Zhang, "A PCA classifier and its application in vehicle-detection," in *Proc. Int. Joint Conf. Neural Netw.*, 2001, vol. 1, pp. 600–604.

[15] W. Efenberger, Q.-H. Ta, L. Tsinas, and V. Graefe, "Automatic recognition of vehicles approaching from behind," in *Proc. Intell. Vehicles Symp.*, 1992, pp. 57–62.

[16] A. Watanabe, M. Andoh, N. Chujo, and Y. Harata, "Neocognitron capable of position detection and vehicle recognition," in *Proc. IEEE IJCNN*, 1999, vol. 5, pp. 3170–3173.

[17] W. Wu, Q. S. Zhang, and M. Wang, "A method of vehicle classification using models and neural networks," in *Proc. IEEE VTC*, 2001, vol. 4, pp. 3022–3026.

[18] X. Limin, "Vehicle shape recovery and recognition using generic models," in *Proc. World Congr. Intell. Control Autom.*, 2002, pp. 1055–1059.

[19] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia, "Event detection and analysis from video streams," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 873–889, Aug. 2001.

[20] Y. Shan, H. S. Sawhney, and R. Kumar, "Unsupervised learning of discriminative edge measures for vehicle matching between non overlapping cameras," in *Proc. Conf. CVPR*, 2005, vol. 1, pp. 894–901.

[21] Y. Guo, S. Hsu, Y. Shan, H. Sawhney, and R. Kumar, "Vehicle fingerprinting for reacquisition & tracking in videos," in *Proc. IEEE Conf. CVPR*, 2005, vol. 2, pp. 761–768.

[22] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proc. IEEE Conf. CVPR*, 2003, vol. 2, pp. 264–271.

[23] G. D. Sullivan, K. D. Baker, A. D. Worrall, C. I. Attwood, and P. M. Remagnino, "Model-based vehicle detection and classification using orthographic approximations," *Image Vis. Comput.*, vol. 15, no. 8, pp. 649–654, Aug. 1997.

[24] D. Koller, K. Daniilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *Int. J. Comput. Vis.*, vol. 10, no. 3, pp. 257–281, Jun. 1993.

[25] D. Freedman and M. W. Turek, "Illumination-invariant tracking via graph-cuts," in *Proc. IEEE Conf. CVPR*, 2005, vol. 2, pp. 10–17.

[26] R. Kumar, H. S. Sawhney, and A. R. Hanson, "3D model acquisition from monocular image sequences," in *Proc. CVPR*, 1992, pp. 209–215.

[27] J. M. Ferryman, A. D. Worrall, and S. J. Maybank, "Learning enhanced 3D models for vehicle tracking," in *Proc. Brit. Mach. Vis. Conf.*, 1998, pp. 873–882.

[28] P. M. Q. Aguiar and J. M. F. Moura, "Fast 3D modeling from video," in *Proc. IEEE 3rd Workshop Multimedia Signal Process.*, 1999, pp. 289–294.

[29] S. Noronha and R. Nevatia, "Detection and description of buildings from multiple aerial images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 5, pp. 501–518, May 2001.

[30] Z. W. Kim and R. Nevatia, "Expandable Bayesian networks for 3D object description from multiple views and multiple mode inputs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 6, pp. 769–774, Jun. 2003.

[31] P. Claes, D. Vandermeulen, L. Van Gool, and P. Suetens, "Partial surface integration based on variational implicit functions and surfaces for 3D model building," in *Proc. Int. Conf. 3D Dig. Imag. Model.*, 2005, pp. 31–38.

[32] P. K. Allen, A. Troccoli, B. Smith, S. Murray, I. Stamos, and M. Leordeanu, "New methods for digital modeling of historic sites," *IEEE Comput. Graph. Appl.*, vol. 23, no. 6, pp. 32–41, Nov./Dec. 2003.

[33] K. Jankowska, T. Krzyzynski, and J. Santos-Victor, "Fusion of 3D models constructed on the basis of omnidirectional images," in *Proc. 3rd Int. Workshop Robot Motion Control*, 2002, pp. 363–368.

[34] Y. K. Ho and C. S. Chua, "3D model building," in *Proc. 7th Int. Conf. Image Process. Appl.*, 1999, vol. 1, pp. 270–274.

[35] A. R. Várkonyi-Kóczy, "Autonomous 3D model reconstruction and its intelligent applications in vehicle system dynamics—Part I: Theory," in *Proc. IEEE Int. Symp. Intell. Syst. Informat.*, Aug. 24–25, 2007, pp. 13–18.

[36] A. R. Várkonyi-Kóczy, "Autonomous 3D model reconstruction and its intelligent applications in vehicle system dynamics—Part II: Applications," in *Proc. IEEE Int. Symp. Intell. Syst. Informat.*, Aug. 24–25, 2007, pp. 19–24.

[37] A. Aulcair, L. Cohen, and N. Vincent, "A robust approach for 3D cars reconstruction," in *Proc. SCIA*, vol. 4522, *LNCS*, 2007, pp. 183–192.

[38] W. Zhang, Q. M. J. Wu, X. Yang, and X. Fang, "Multilevel framework to detect and handle vehicle occlusion," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 161–174, Mar. 2008.

[39] N. K. Kanhere and S. T. Birchfield, "Real-time incremental segmentation and tracking of vehicles at low camera angles using stable features," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 148–160, Mar. 2008.

[40] I. Urazghildiiev, R. Ragnarsson, P. Ridderström, A. Rydberg, E. Öjefors, K. Wallin, P. Enochsson, M. Ericson, and G. Löfqvist, "Vehicle classification based on the radar measurement of height profiles," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 245–253, Jun. 2007.

[41] W. W. L. Lam, C. C. C. Pang, and N. H. C. Yung, "Vehicle-component identification based on multiscale textural couriers," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 4, pp. 681–694, Dec. 2007.

[42] B. Li, R. Chellappa, Q. Zheng, and S. Z. Der, "Model-based temporal object verification using video," *IEEE Trans. Image Process.*, vol. 10, no. 6, pp. 897–908, Jun. 2001.

[43] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, pp. 1–45, 2006, Art. 13.

[44] A. Ottlik and H.-H. Nagel, "Initialization of model-based vehicle tracking in video sequences of inner-city intersections," *Int. J. Comput. Vis.*, vol. 80, no. 2, pp. 211–225, Nov. 2008.

[45] H. Dahlkamp, H.-H. Nagel, A. Ottlik, and P. Reuter, "A framework for model-based tracking experiments in image sequences," *Int. J. Comput. Vis.*, vol. 73, no. 2, pp. 139–157, Jun. 2007.

[46] Z. W. Kim and J. Malik, "Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking," in *Proc. IEEE ICCV*, 2003, pp. 524–531.

[47] J. Lou, T. Tan, W. Hu, H. Yang, and S. J. Maybank, "3-D model-based vehicle tracking," *IEEE Trans. Image Process.*, vol. 14, no. 10, pp. 1561–1569, Oct. 2005.

[48] W. Hu, X. Xiao, D. Xie, T. Tan, and S. Maybank, "Traffic accident prediction using 3-D model-based vehicle tracking," *IEEE Trans. Veh. Technol.*, vol. 53, no. 3, pp. 677–694, May 2004.

[49] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Auton. Robots*, vol. 26, no. 2/3, pp. 123–139, Apr. 2009.

[50] M. J. Leotta and J. L. Mundy, "Predicting high resolution image edges with a generic, adaptive, 3-D vehicle model," in *Proc. IEEE CVPR*, 2009, pp. 1311–1318.

[51] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[52] S. Nadimi and B. Bhanu, "Physical models for moving shadow and object detection in video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1079–1087, Aug. 2004.

[53] N. Ghosh and B. Bhanu, "Incremental vehicle 3-D modeling from video," in *Proc. IEEE ICPR*, 2006, vol. 3, pp. 272–275.

[54] C.-C. R. Wang and J.-J. J. Lien, "Automatic vehicle detection using local features—A statistical approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 83–96, Mar. 2008.

[55] M. Sarvi and M. Kuwahara, "Using ITS to improve the capacity of freeway merging sections by transferring freight vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 4, pp. 580–588, Dec. 2008.

[56] Q. J. Kong, Z. Li, Y. Chen, and Y. Liu, "An approach to urban traffic state estimation by fusing multisource information," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 499–511, Sep. 2009.

[57] W.-Y. Shieh, T.-H. Wang, Y.-H. Chou, and C.-C. Huang, "Design of the radiation pattern of infrared short-range communication systems for electronic-toll-collection applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 548–558, Sep. 2008.

[58] H. G. Jung, Y. H. Cho, P. J. Yoon, and J. Kim, "Scanning laser radar-based target position designation for parking aid system," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 406–424, Sep. 2008.

Nirmalya Ghosh received the B.Tech. and M.S. degrees from the Indian Institute of Technology, Kharagpur, India, in 1998 and 2002, respectively, and the Ph.D. degree from the University of California, Riverside, in 2007, all in eletrical engineering.

He is currently a Research Associate with the Department of Pediatrics, Loma Linda University, Loma Linda, CA. His research interests include image and video processing, 3-D model building, medical image processing, intelligent controls and process monitoring, machine learning, and pattern recognition.

Bir Bhanu (S'72–M'82–SM'87–F'96) received the S.M. and E.E. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, the Ph.D. degree in electrical engineering from the Image Processing Institute, University of Southern California, Los Angeles, and the MBA degree from the University of California, Irvine.

He is the Distinguished Professor of Electrical Engineering and serves as the Founding Director of the interdisciplinary Center for Research in Intelligent Systems (CRIS) at the University of California at Riverside (UCR). He was the founding Professor of electrical engineering and served as its first chair (1991–1994). He has been the Cooperative Professor of Computer Science and Engineering (since 1991), Bioengineering (since 2008), Mechanical Engineering (since 2008), and the Director of Visualization and Intelligent Systems Laboratory (since 1991). Previously, he was a Senior Honeywell Fellow with Honeywell Inc., Minneapolis, MN. He has been with the faculty of the Department of Computer Science, University of Utah, Salt Lake City, and with Ford Aerospace and Communications Corporation, Newport Beach, CA; INRIA-France; and IBM San Jose Research Laboratory, San Jose, CA. He has been the principal investigator of various programs for the National Science Foundation, the Defense Advanced Research Projects Agency (DARPA), the National Aeronautics and Space Administration, the Air Force Office of Scientific Research, the Office of Naval Research, the Army Research Office, and other agencies and industries in the areas of video networks, video understanding, video bioinformatics, learning and vision, image understanding, pattern recognition, target recognition, biometrics, autonomous navigation, image databases, and machine-vision applications. He is a coauthor of the books *Computational Learning for Adaptive Computer Vision* (forthcoming), *Human Ear Recognition by Computer* (Springer-Verlag, 2008), *Evolutionary Synthesis of Pattern Recognition Systems* (Springer-Verlag, 2005), *Computational Algorithms for Fingerprint Recognition* (Kluwer, 2004), *Genetic Learning for Adaptive Image Segmentation* (Kluwer, 1994), and *Qualitative Motion Understanding* (Kluwer, 1992), and the coeditor of the book *Computer Vision Beyond the Visible Spectrum* (Springer-Verlag, 2004). He is the holder of 11 U.S. and international patents. He has more than 350 reviewed technical publications.

Dr. Bhanu is a Fellow of the IEEE Computer Society, the American Association for the Advancement of Science, the International Association of Pattern Recognition, and the International Society for Optical Engineering. He has been on the editorial board of various journals and has edited special issues of several IEEE TRANSACTIONS (PATTERN ANALYSIS AND MACHINE INTELLIGENCE; IMAGE PROCESSING; SYSTEMS, MAN AND CYBERNETICS—PART B; ROBOTICS AND AUTOMATION; INFORMATION FORENSICS AND SECURITY) and other journals. He was the General Chair for the IEEE Conference on Computer Vision and Pattern Recognition, the IEEE Conference on Advanced Video and Signal-based Surveillance, the IEEE Workshops on Applications of Computer Vision, the IEEE Workshops on Learning in Computer Vision and Pattern Recognition; the Chair for the DARPA Image Understanding Workshop and the IEEE Workshops on Computer Vision Beyond the Visible Spectrum and Multi-Modal Biometrics. He was the recipient of best conference papers and outstanding journal paper awards and the industrial and university awards for research excellence, outstanding contributions, and team efforts.