Transportation Research Board Annual Meeting, Washington, D.C., January 2008. **TRB Paper Number:**
Word count: 5100 + 7 figures + 2 tables

Automatic Camera Calibration Using Pattern Detection for Vision-Based Speed Sensing

Neeraj K. Kanhere
Department of Electrical and Computer Engineering
207-A Riggs Hall
Clemson University, Clemson, SC 29634
Phone: (864) 650-4844, FAX: (864) 656-5910
E-mail: nkanher@clemson.edu

Stanley T. Birchfield
Department of Electrical and Computer Engineering
207-A Riggs Hall
Clemson University, Clemson, SC 29634
Phone: (864) 656-5912, FAX: (864) 656-5910
E-mail: stb@clemson.edu

Wayne A. Sarasua
Department of Civil Engineering
312 Lowry Hall, Box 340911
Clemson University, Clemson, SC 29634
Phone: (864) 656-3318, FAX: (864) 656-2670
E-mail: sarasua@clemson.edu

August 1, 2007

ABSTRACT

Vision-based automatic traffic monitoring systems require a calibrated camera in order to measure the speeds of tracked vehicles. Typically this calibration is done by hand. We present an automatic technique to calibrate the camera for typical viewpoints on highways using a real-time boosted cascade vehicle detector (BCVD). Image processing is used to estimate the two vanishing points, from which the camera height, focal length, and pan and tilt angles are calculated. The key contribution of the proposed approach is its applicability to a wide variety of environmental conditions. The technique does not rely upon background subtraction, nor does it require scene-specific features such as pavement markings. As a result, it is unaffected by the presence of shadows, adverse weather conditions, headlight reflections, lack of ambient light, or spillover caused by low-mounted cameras. Speed estimation within 10% of ground truth is shown for sequences obtained during daylight, nighttime, and rain, and including shadows and severe occlusion.

INTRODUCTION

Vision-based processing is becoming an increasingly popular approach to solving the problem of vehicle tracking for automatic traffic surveillance applications (*1*, *2*, *3*, *4*). Camera calibration is an essential step in such systems to measure speeds, and it often improves the accuracy of tracking techniques for obtaining vehicles counts, as well. Typically, calibration is performed by hand, or at least semi-automatically. For example, an algorithm for interactive calibration of a Pan-Tilt-Zoom (PTZ) camera has been proposed in (*5*). Bas and Crisman (*6*) use the known height and the tilt angle of the camera for calibration using a single set of parallel lines (along the road edges) drawn by the user, while Lai (7) removes the restriction of known height and tilt angle by using an additional line of known length perpendicular to the road edges. The technique of Fung et al. (*8*), which uses the pavement markings and known lane width, is robust against small perturbations in the markings, but it requires the user to draw a rectangle formed by parallel lane markings in adjacent lanes. The problem of ill-conditioned vanishing points (i.e., parallel lines in the world appearing parallel in the image) has been addressed by He et al. (*9*) using known length and width of road lane markings. Additional techniques for manual camera calibration are described in (*1*, *3*, *4*).

Recently the alternative of automatic camera calibration has gained some attention. Automatic calibration would not only reduce the tediousness of installing fixed cameras, but it would also enable the use of PTZ cameras without manually recalibrating whenever the camera moves. Dailey et al. (*10*) relate pixel displacement to real-world units by fitting a linear function to scaling factors obtained using a known distribution of typical length of vehicles. Sequential image frames are subtracted, and vehicles are tracked by matching the centroids of the resulting blobs. At low camera heights, the resulting spillover and occlusion cause blobs to be merged, which renders such tracking ineffective. In follow-up research, Schoepflin and Dailey (*11*) dynamically calibrate PTZ cameras using lane activity maps which are computed by frame-differencing. As noted in their paper, spillover is a serious problem for moderate to large pan angles, and this error only increases with low camera heights. In our experience we have found that estimating lanes using activity maps is impossible with pan angles as small as 10° when the camera is placed 20 feet above the ground, due to the large amount of spillover and occlusion that occur. In an alternate approach, Song et al. (*12*) use edge detection to find the lane markings in the static background image, from which the vanishing point is estimated by assuming that the camera height and lane width are known in advance. The method requires the lane markings to be visible, which may not be true under poor lighting or weather conditions. In addition, estimating the static background is not always possible when the traffic is dense, it requires time to acquire a good background image, and background subtraction does not work well at low camera heights due to occlusion and spillover, as noted above.

In this paper we present a system that overcomes several of the limitations of previous approaches. The approach does not require pavement markings or prior knowledge of the camera height or lane width; it is unaffected by spillover, occlusion, and shadows; and it works in dense traffic and different lighting and weather conditions. The key to the success of the system is a boosted cascade vehicle detector (BCVD) that extracts vehicles from a single image using the surrounding intensity pattern, without using any motion information. Since vehicles are detected and tracked using their intensity patterns in the image, the system does not suffer from the well-known drawbacks of background subtraction or frame differencing. The technique uses the vehicle trajectories in the image and the intensity gradient along the vehicle windshield to compute the two vanishing points in the image, from which the camera parameters (height, focal length, and pan and tilt angles) are estimated.

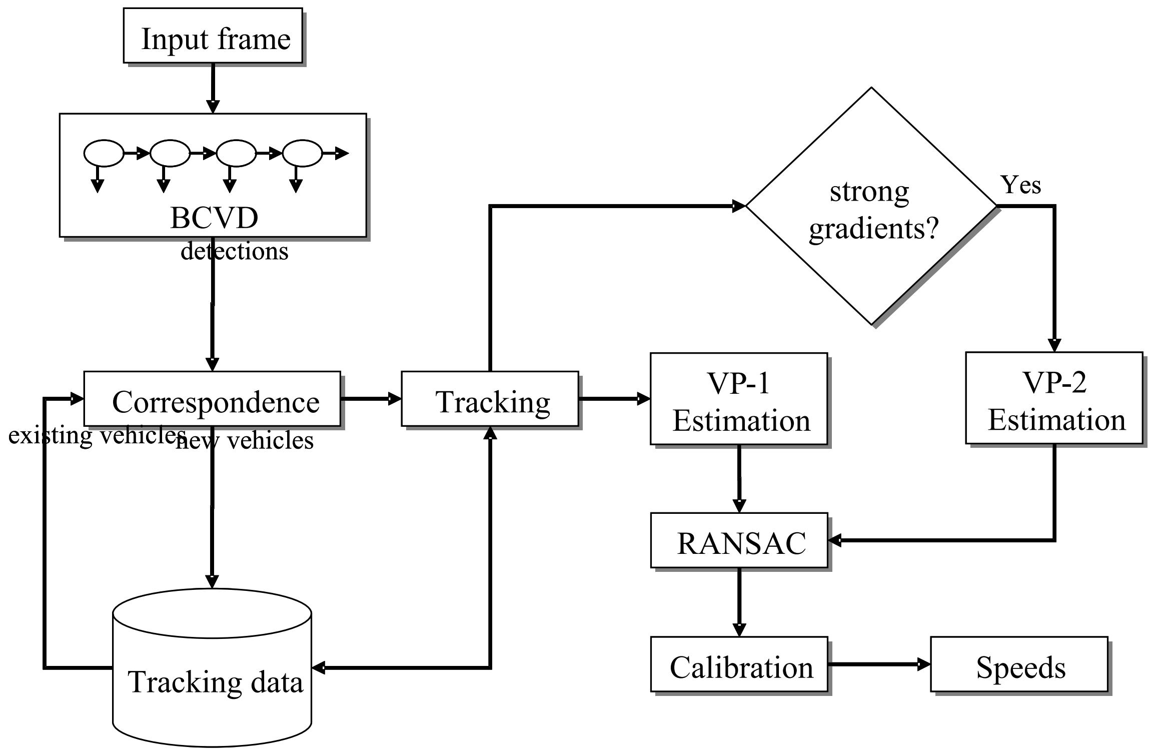
APPROACH

As in (*7*, *11*, *12*), we adopt a pinhole camera model with the following assumptions: the road surface is flat, the roll angle of the camera is zero, the aspect ratio of the pixels is unity, and the principal point (the intersection of the optical axis and the image plane) is the image center. With these assumptions, four parameters are needed to map between pixel distances (measured in the image) and corresponding distances on the road (measured in Euclidean world units): Focal length (f), tilt angle (φ), pan angle (θ), and height of the camera measured from the road surface (h).

Figure 1 presents an overview of the system. The bulk of the processing is performed by a boosted cascade vehicle detector (BCVD), which is used to detect and track vehicles. The resulting vehicle tracks are then used to estimate the first vanishing point in the direction of travel, while strong gradients near vehicle windshields (in daytime) or the lines joining the two headlights (at night) are used to compute the second vanishing point in the direction perpendicular to the direction of travel. The Random Sample Consensus (RANSAC) algorithm (13) is used to eliminate outliers resulting from noise and/or image compression artifacts. From the vanishing points, the camera is calibrated, which then enables the speed of vehicles to be computed by mapping pixel coordinates to world distances. The only parameter of the system is the mean vehicle width, which is assumed to be 7 feet (*14*).

One useful characteristic of our approach is that the system is calibrated incrementally. In other words, only two images of a single vehicle are needed in principle to calibrate the system, thus providing a nearly instantaneous solution to the problem. This unique behavior eliminates the delay inherent in background subtraction techniques, which makes the system amenable for use by PTZ cameras whose parameters are continually changing. In practice, although we use the first vehicle to obtain initial calibration parameters, we refine those parameters over time as more vehicles are detected and tracked in order to obtain more accurate estimates. Additional advantages of the approach include its immunity to shadows (Note that Dailey et al. (*10*) observed more than 10% error in mean speed estimates due to shadows), as well as its insensitivity to spillover and/or dense traffic, since vehicles are detected using a discriminative set of features as opposed to simple foreground blobs.

FIGURE 1 Overview of the system.



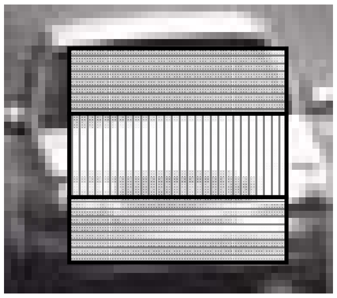
Block diagram of the proposed system for automatic camera calibration.

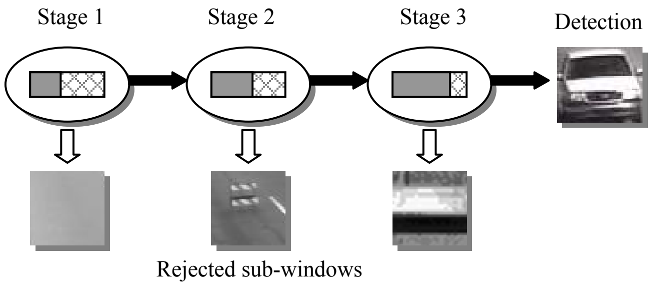
Boosted Cascade Vehicle Detector

The problem of pattern classification has been studied extensively for many years, giving rise to a variety of approaches such as neural networks, support vector machines (SVMs), and Bayesian classifiers. A relatively new approach using a cascade of simple features to detect patterns in images was recently developed by Viola and Jones (*15*). Their approach is illustrated in Figure 2. Each image sub-window is passed through a series of tests of increasing difficulty, known as a *cascade*. The goal of each stage in the cascade is to evaluate the sub-window using a set of image features to decide whether to reject the sub-window as containing the object of interest. Subsequent stages perform more detailed analyses using larger and more discriminating sets of features, with each stage trained to achieve a high hit rate (e.g., 99%) and a liberal false alarm rate (e.g., 50%). Sub-windows in the image which are easily distinguishable as non-vehicles (e.g., an image patch with little or no texture) are discarded in the initial stages of the cascade, resulting in faster processing, so that the complete set of features needs to be evaluated for only the small fraction of sub-windows that reach the final stage of the cascade. The training process ensures that the classification errors in each stage are independent of each other.

The Viola-Jones algorithm achieves real-time processing not only with the cascade architecture, but also because it uses simple image difference features that are quickly computed using an integral image. The features used in (*15*) are simply arithmetic additions and subtractions of pixel intensities in a detection window. An example of such a feature is shown in Figure 2 (a) where the value of a feature is computed by subtracting the sum of pixel intensities in the top and the bottom regions (horizontal bars) from the sum of pixel intensities in the middle region (vertical bars). Given a set of labeled training images (vehicles and non-vehicles), the training process first finds a feature (from a large pool of rectangular features) and a corresponding threshold on the value of the feature that performs best on the training data. A single feature in essence acts as a *weak classifier* whose decision is at least slightly better than random chance. The idea behind boosting is to combine several such weak classifiers in a way such that the final *strong classifier* meets the performance requirements. After training, vehicles are detected by sliding the strong classifier over the input image and computing the decision (vehicle or non-vehicle) at each sub-window in the image. To detect vehicles at different scales, the feature set (and in effect the detection window) is scaled (rather than the more traditional approach of resampling of the input image), which further reduces the computational load.

FIGURE 2 Cascade of simple features.





(a) (b)

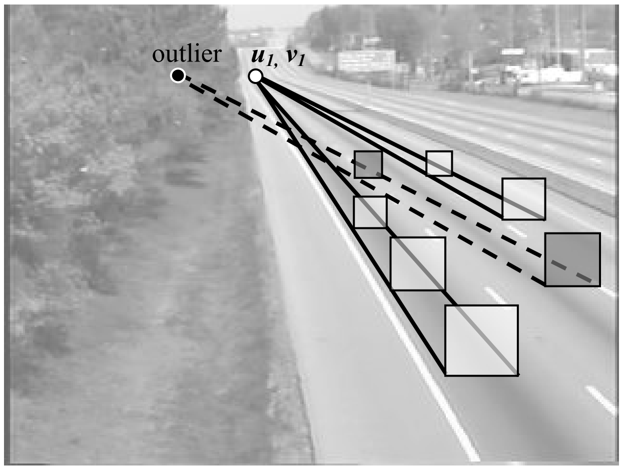
(a) Example of simple feature. (b) Cascade architecture for fast detection.

Correspondence and Tracking

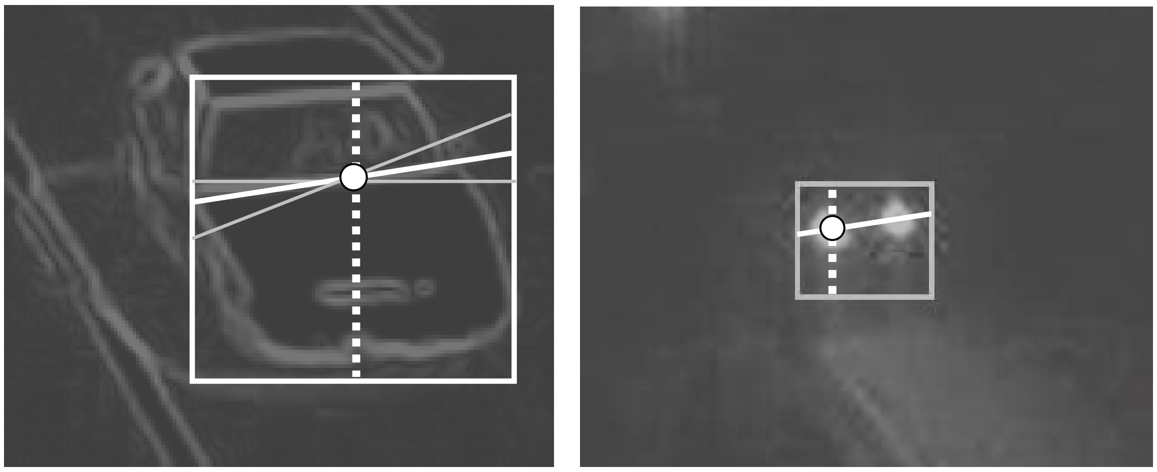
Each image of the video sequence is scanned exhaustively at multiple scales by the BCVD to detect vehicles. (An entire image can be scanned in a fraction of a second using a standard computer.) The output of the BCVD is a rectangle for each detected vehicle, and the midpoint along the bottom edge of the rectangle is retained as the location of the vehicle for the purpose of computing proximity to other vehicles. Vehicles from the previous image frame are tracked by searching among nearby detections in the current image frame. In case a match is not found, the vehicle is flagged as missing and its location is updated by means of a standard template matching mechanism using normalized cross-correlation. If a vehicle is missing for several consecutive frames, it is discarded for the lack of sufficient evidence. Meanwhile, new vehicles are initialized for all the detections that did not yield a match. This straightforward tracking procedure augments the position information of the vehicles with their image trajectories.

Estimating Vanishing Points

FIGURE 3 Estimation of vanishing points.



(a)



(b) (c)

(a) Vanishing point in the direction of travel is estimated using vehicle tracks. Tracking errors or a vehicle changing lanes (dark rectangles) might lead to outliers. (b) Gradient magnitudes are used to find a hinge point (shown as a white circle) followed by slope estimation during day light conditions. (c) Raw pixel intensities are used to estimate second vanishing point using headlights of vehicles.

Lines which are parallel to each other in real world generally do not appear parallel in the image (except when they are parallel to the image plane). As an example, consider an aerial photograph of rail-road tracks with the camera looking straight down. The tracks will appear parallel to each other in the image. If another image is taken standing in the middle of the tracks and pointing the camera straight ahead (camera looking towards horizon), the tracks will appear to meet at a finite point in the image plane. This point of intersection is called a vanishing point. A vanishing point is defined only by the direction of lines, in other words, all parallel lines in a particular direction will appear to converge at a single unique location in the image.

The vanishing point $\boldsymbol{p_1} = (\boldsymbol{u_1}, \boldsymbol{v_1})$ in the direction of travel is estimated using vehicle tracks. We fit a line passing through bottom-left and bottom-right image coordinates of all the detection windows for a vehicle. Estimating the vanishing point directly from the vehicle tracks avoids using computationally expensive Hough transform (*16*). Figure 3 (a) illustrates a scenario where a vehicle changing lanes (represented by darker rectangle) results into an outlier. In addition, tracking and localization errors can lead to outliers. We use RANSAC for removing the bias in the estimation of vanishing points resulting from outliers.

To estimate the vanishing point $\boldsymbol{p_2} = (\boldsymbol{u_2}, \boldsymbol{v_2})$ in the direction perpendicular to traffic-flow, we employ strong image gradients found on light colored vehicles. Apparent slope of a line in an image (corresponding to a line in real world along the direction perpendicular to traffic-flow) is inversely proportional to its distance from the camera. Estimating $\boldsymbol{p_2}$ as the intersection of two lines in its direction is very sensitive to measurement errors. With the assumption that the camera has zero roll, we can find as the intersection of $\boldsymbol{v} = \boldsymbol{v_1}$ and a line corresponding to the perpendicular direction. We use the detection window that is closest to the camera (close to the bottom edge of an image) to search for a *hinge point*, which is a point of maximum gradient magnitude and lies along the vertical axis passing through the center of the window (along the dashed line). Next, we search for a line passing through the hinge point and having a slope that maximizes the sum of gradients along that line. In Figure 3(b), the white circle indicates the location of the hinge point. Among the three example candidates, the line that coincides with the edge of the windshield of the vehicle (shown as a solid line) is used to compute $\boldsymbol{p_2}$. In case of absence of any ambient light, we use headlights to estimate $\boldsymbol{p_2}$. The hinge point is found along a vertical axis shifted to left by quarter of detection window width as shown in Figure 3(c). Note that raw pixel intensities are used in this case as opposed to gradient magnitude image used earlier.

Calibration

Location of a vanishing point in an image is independent of the camera placement and depends only on the intrinsic camera parameters and its orientation (*17*). In our case, vanishing points are independent of the camera height h and depend on focal length f, tilt angle φ and pan angle θ. Once the two vanishing points $\boldsymbol{p_1} = (\boldsymbol{u_1}, \boldsymbol{v_1})$ and $\boldsymbol{p_2} = (\boldsymbol{u_2}, \boldsymbol{v_2})$ have been estimated, with $\boldsymbol{v_1} = \boldsymbol{v_2}$, the parameters f, φ and θ can be computed as follows (see the Appendix):

$$f = \sqrt{-\left(v_1^2 + u_1 u_2\right)}$$

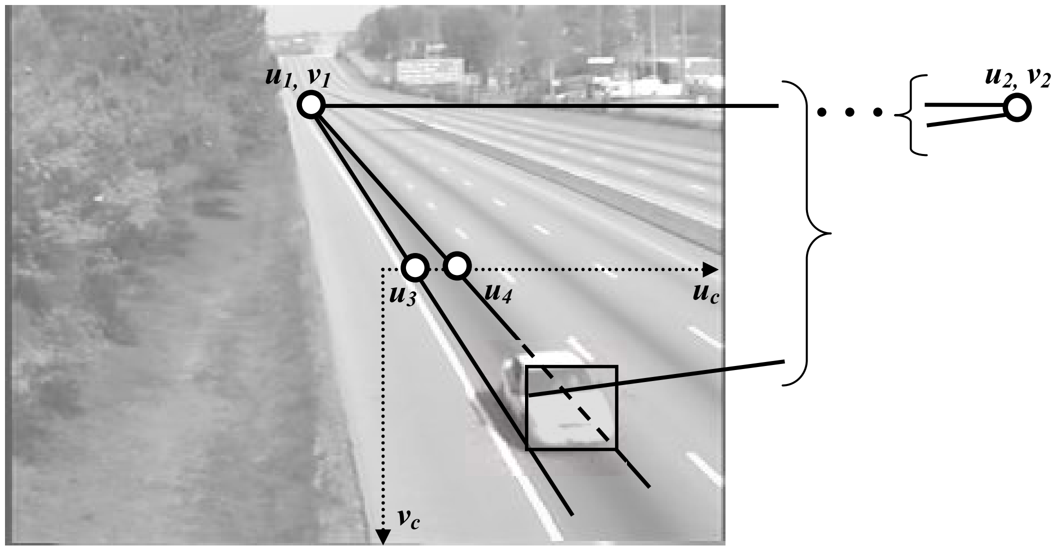
$$\phi = \tan^{-1}\left(\frac{-v_1}{f}\right)$$

$$\theta = \tan^{-1}\left(\frac{-u_1 \cos\phi}{f}\right)$$

To compute the height h of the camera, we need to locate two points along the horizontal axis in the image. As shown in Figure 4, $\boldsymbol{u_3}$ and $\boldsymbol{u_4}$ are u-axis intercepts of lines connecting $\boldsymbol{p_1}$ with the bottom-left and bottom-right points of the detection window respectively. Finally, the height is computed using an assumed average width of a car (*14*).

$$h = \frac{wf\sin\phi}{(u_4 - u_3)\cos\theta}$$

FIGURE 4 Image measurements.



Calibration parameters are computed using the four points shown above and from assumed mean width of a vehicle.

Measuring Speeds

Once the camera has been calibrated, the pixel location of a vehicle in the image (u, v) can be mapped into a location on the road (x, y) using following equations:

$$x = \frac{uh}{v\cos\phi + f\sin\phi}$$
$$y = \frac{h(f - v\tan\phi)}{v + f\tan\phi}$$

The distance traveled by a vehicle between two arbitrary image frames can be easily computed using the above relations. The speed of a vehicle is computed using the distance traveled, the corresponding number of frames, and the frame rate (FPS) of the camera. Interested readers can find the derivations for all the above equations in the appendix.

EXPERIMENTAL RESULTS

We used the Intel OpenCV (18) library to train two vehicle detectors (BCVDs), one for daytime and one for night, using the two training sequences shown in Figure 5 (a) and (b). At run time, the system automatically selects the proper detector (day or night) based on the average pixel intensity in the images. To test the system, we captured four image sequences, three during daylight conditions and one at night, using an inexpensive off-the-shelf web camera (Logitech

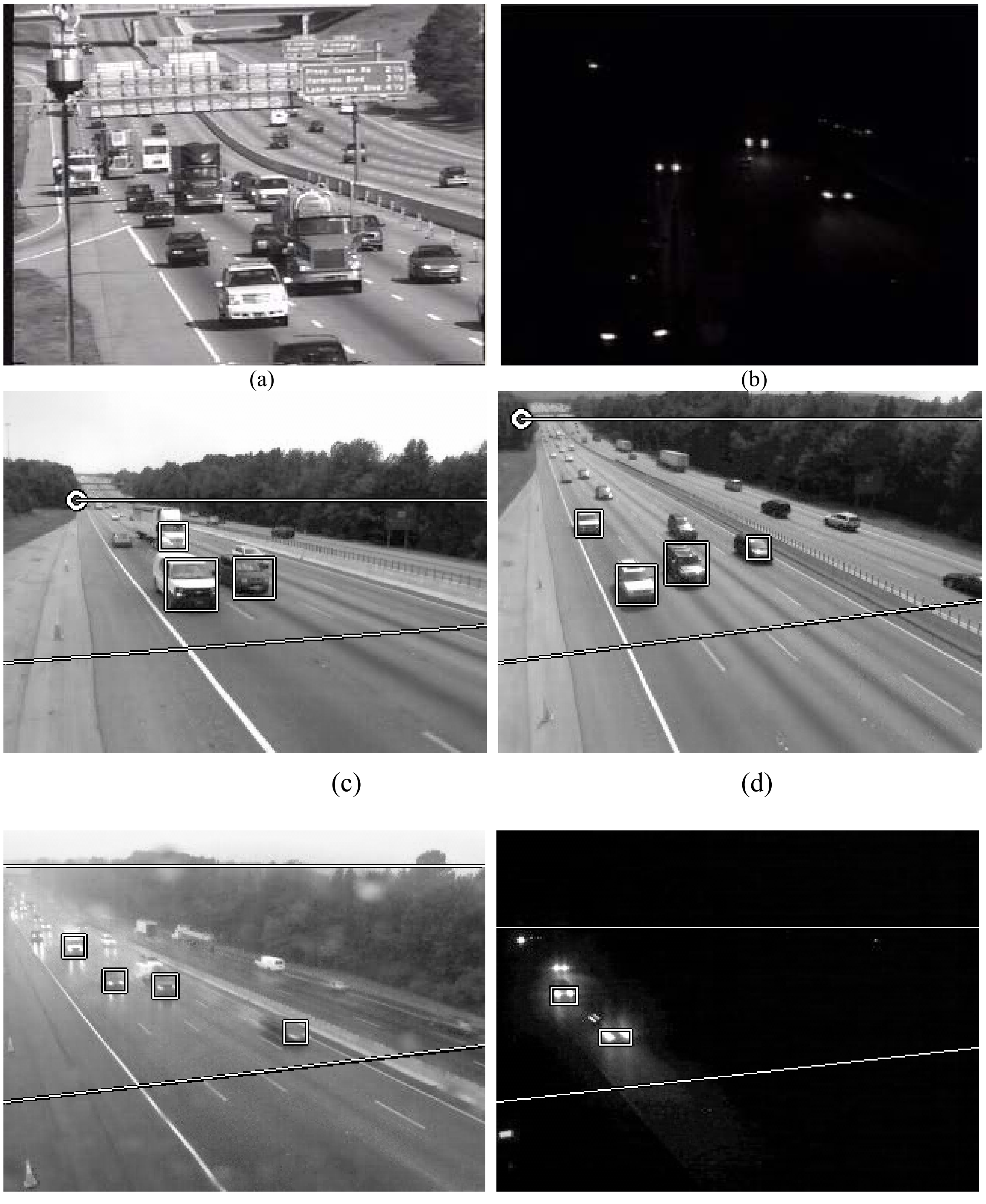
Orbitz) mounted at the top of an adjustable pole. An image from each sequence is shown in Figure 5 (c)-(e). The images were captured at 15 frames per second at 320x240 pixel resolution. Note that we used different cameras for capturing the training and test sequences, and that the cameras were not placed in the same location, thus demonstrating the robustness of the system.

Figure 5 also shows the results overlaid on the images. The rectangles outline the detected vehicles; the false negatives are not a problem since our goal is mean speed rather than vehicle counts. The white circle indicates the first vanishing point, which is only visible in two of the four test sequences. The second vanishing point is very far from the image and is given by the intersection of the horizon line and the other line drawn.

The sequences were approximately 10 minutes long each. A radar was used to compare the mean speed over the entire sequence for three of the sequences, with the results displayed in the table below. Treating the radar as ground truth, the error of our system ranged from 3 to 6 mph, with a slightly greater standard deviation than the radar. Figure 6 shows the error in the distance estimate (displayed as a percentage) versus the amount of data that the algorithm was allowed to use. As mentioned previously, the algorithm instantaneously yields initial estimate, which improves over time as more information is gathered. In two of the sequences the estimate stabilized after only ten vehicles, while the poor weather conditions of the third sequence caused the estimate to require more data.

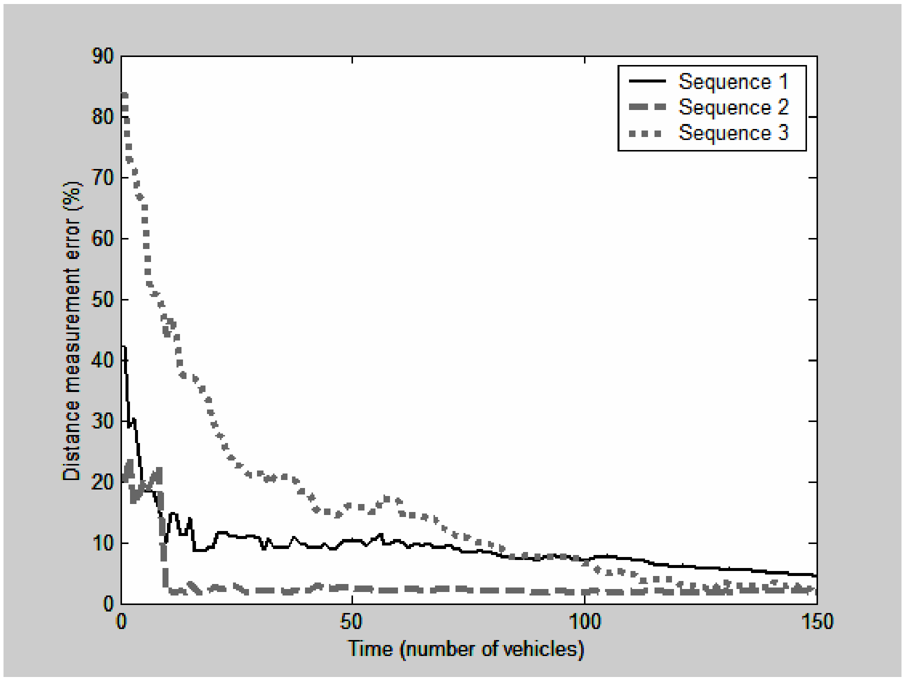
Table 1 shows the accuracy of the estimation of the camera parameters for the four sequences. We computed the accuracy by comparing with camera parameters obtained using the same equations but with hand-labeled vanishing points. In all cases the error is less than 10%. Table 2 displays the speed error for twenty individual vehicles in each of the four sequences. The average error ranges from 3 to 5 mph.

FIGURE 5 Training and Test Sequences.



(a) Training sequence 1. (b) Training sequence 2. (c)-(f) Four test sequences. (c) Sequence 1, *h* = 15 feet, clear day. (d) Sequence 2, *h* = 30 feet, clear day. (e) Sequence 3, *h* = 30 feet, rain with headlight reflections. (f) Sequence 4, *h* = 20 feet, night time, no ambient lighting.

FIGURE 6 Distance error.



Error in measuring known distances decreases over time as more vehicles are detected and tracked.

TABLE 1 Accuracy of parameters and mean speed.

	Sequence 1		**Sequence 2**		**Sequence 3**		**Sequence 4**	
	Manual	**Algorithm**	**Manual**	**Algorithm**	**Manual**	**Algorithm**	**Manual**	**Algorithm**
f (pixels)	376.21	366.83	389.43	382.26	387.04	411.06	382.76	380.16
φ (degrees)	7.12 ^{o}	7.44^o	15.21 ^{o}	14.89 ^{o}	12.82 ^{o}	11.53 ^{o}	23.14 ^{o}	23.77 ^{o}
θ (degrees)	14.97 ^{o}	16.76 ^{o}	19.76 ^{o}	20.05 ^{o}	24.27 ^{o}	22.34 ^{o}	7.83 ^{o}	8.25 ^{o}
h (feet)	15	14.2	30	29.69	30	28.83	20	18.62
	Sequence 1		**Sequence 2**		**Sequence 3**			
	Radar	**Algorithm**	**Radar**	**Algorithm**	**Radar**	**Algorithm**		
μ	61.81	63.92	62.22	61.62	54.3	51.66		
σ	4.42	5.97	3.77	4.78	3.7	5.12		
N	187	520	235	491	196	416		

Accuracy of the estimated parameters compared with parameters computed manually. *f* is the focal length, *φ* is the tilt angle, *θ* is the pan angle, *h* is the camera height. μ, σ and *N* are mean speed for the entire sequence, standard deviation of speeds and number of observations used for computation.

TABLE 2 Accuracy of speed estimates.

Sequence 1					**Sequence 2**			
Vehicle Number	**Lane**	**Measured Speed**	**Algorithm Speed**		**Vehicle Number**	**Lane**	**Measured Speed**	**Algorithm Speed**
145	2	57	53		30	3	64	63
185	2	57	55		84	2	61	58
191	2	51	53		133	1	59	56
254	2	64	63		135	1	59	57
276	2	64	63		246	2	57	57
314	1	64	67		272	3	64	64
326	2	57	55		276	2	64	63
339	2	51	62		318	3	49	62
356	1	64	63		374	3	67	65
357	2	57	61		375	2	55	50
386	1	64	63		379	2	55	56
402	2	51	50		399	2	61	62
407	1	57	54		419	1	59	57
442	2	51	56		431	4	67	64
447	1	64	73		458	3	64	62
472	2	57	56		464	2	57	56
504	1	64	65		524	3	59	59
505	2	73	61		543	2	67	65
507	1	64	61		601	4	61	62
513	2	64	65		608	2	64	62
Mean error (mph)			**3.5**		**Mean error (mph)**			**2.25**
Sequence 3					**Sequence 4**			
129	1	59	54		1	2	45	48
130	3	55	55		5	2	55	53
154	2	57	52		8	2	46	42
164	3	57	53		17	1	57	59
176	3	59	54		21	2	45	43
193	2	55	50		25	1	46	48
202	1	64	57		34	1	59	59
205	3	59	60		37	1	55	51
213	2	59	54		39	1	48	43
239	2	57	53		42	1	53	50
289	2	57	51		46	1	46	43
354	4	61	57		52	1	57	52
373	2	57	50		55	1	53	52
406	2	51	47		59	2	53	49
427	3	53	45		61	1	43	43
444	1	57	50		62	2	57	51
471	3	55	49		64	2	53	56
510	2	46	40		66	1	57	54
551	2	53	49		71	2	57	54
574	3	55	49		72	1	45	44
Mean error (mph)			**4.95**		**Mean error (mph)**			**2.8**

Ground-truth speeds were measured manually by observing the video with the help of markers placed in the scene. Vehicles were chosen at random to compare accuracy of speed estimation.

CONCLUSIONS AND RECOMMENDATIONS

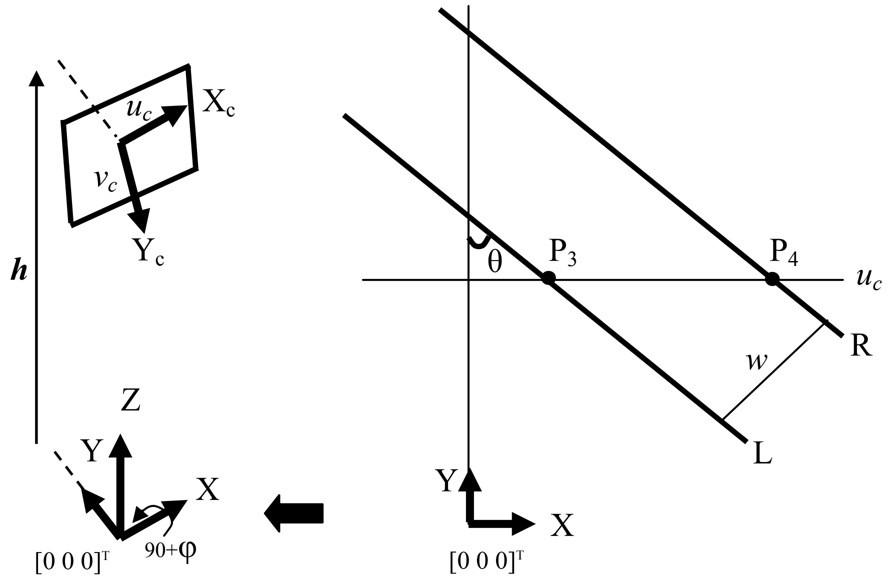
We have presented a method for calibrating a camera along the side of a road using a Boosted Cascade Vehicle Detector (BCVD). The BCVD detects vehicles in images by comparing the 2D intensity patterns with a model acquired during an off-line, one-time training phase. The training does not have to be performed on images captured at the same location or by the same camera as those used at run-time. Because the technique does not use motion information, it overcomes many of the limitations of the common approaches of background subtraction or frame differencing. For example, an estimate is available immediately upon detecting and tracking a single vehicle between two image frames, thus supporting applications such as Pan-Tilt-Zoom (PTZ) cameras in which it may not be feasible to allow the algorithm to learn the background model every time the camera is moved. In addition, the algorithm is insensitive to shadows, spillover, occlusion, and environmental conditions, and it is applicable in daytime or nighttime scenarios. We believe that this work demonstrates the potential for pattern-based detectors to detect and track vehicles in highway scenarios, and that it enhances the usefulness of cameras by obviating the need for tedious manual calibration procedures. Future work involves expanding the technique to work with rear-facing vehicles receding from the camera, augmenting the pattern detector with other modalities to decrease convergence time, and introducing partial calibration when some camera parameters are already known from previous iterations of the algorithm.

ACKNOWLEDGMENT

We would like to thank the James Clyburn Transportation Center at South Carolina State University and the Dwight David Eisenhower Transportation Fellowship Program for supporting this research. We would also like to thank Robert Graham for collecting the video data.

APPENDIX

FIGURE 7 Camera placement.



Relationship between camera and world coordinate frames.

Origin of the World coordinates frame is at the base of the camera (at $[0\ 0\ 0]^T$). Camera is placed at height h and rotated around X axis by an angle of $(90+\varphi)$ to obtain camera coordinate frame (X_c, Y_c, Z_c) so that the optical axis of the camera is at an angle of φ with the horizon. In image plane, u-axis coincides with X_c while v-axis coincides with Y_c.
The optical axis of the camera is at θ angle with respect to traffic lanes. L and R are two arbitrary lines in the direction of travel passing through left and right sides of a vehicle of assumed width w.
We derive the equations for camera parameters using Homogeneous coordinates (*17*). Using homogeneous coordinates allows us to use simpler notation using matrices.

Let **X** be a point in the world with its image as **x**. The relationship between the two is captured by the matrix ***P***.

$$\mathbf{x} = P\mathbf{X} \tag{1}$$

With our assumption that the camera has zero roll, uniform aspect ratio, square pixels and that the principal point coincides with the image center, we get the following expression for ***P*** as explained in (*17*).

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & -f\sin\phi & -f\cos\phi & fh\cos\phi \\ 0 & \cos\phi & -\sin\phi & h\sin\phi \end{bmatrix} \tag{2}$$

Substituting ***P*** in (1) we get expressions that relates world coordinates [x, y, z] of a point in the world with image coordinates [u, v]:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & -f\sin\phi & -f\cos\phi & fh\cos\phi \\ 0 & \cos\phi & -\sin\phi & h\sin\phi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3}$$

Let $p_1 = [su_1, sv_1, s]$ and $p_2 = [su_2, sv_2, s]$ be the vanishing points correspond to a vanishing lines $l_1 = [-tan\theta, 1, 0, 0]$ and $l_2 = [-1, -tan\theta, 0, 0]$ respectively. Vanishing line l_1 is along the direction of travel, while l_2 is perpendicular to the direction of travel.

From equation (1), we get

$$\begin{bmatrix} su_1 \\ sv_1 \\ s \end{bmatrix} = P \begin{bmatrix} -\tan\theta \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

and

$$\begin{bmatrix} su_2 \\ sv_2 \\ s \end{bmatrix} = P \begin{bmatrix} -1 \\ -\tan\theta \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

Expanding the last two equations:

$$u_1 = \frac{-f\tan\theta}{\cos\phi} \qquad v_1 = \frac{-f\sin\phi}{\cos\phi}$$

$$u_2 = \frac{f}{\cos\phi\tan\theta} \qquad v_2 = v_1 = \frac{-f\sin\phi}{\cos\phi}$$

The above equations can be solved to compute f, φ and θ.

$$f = \sqrt{-\left(v_1^2 + u_1 u_2\right)} \qquad \phi = \tan^{-1}\left(\frac{-v_1}{f}\right) \qquad \theta = \tan^{-1}\left(\frac{-u_1 \cos\phi}{f}\right)$$

Substituting *v*=0 in equation (3) yeilds:

$$y_{\{v=0\}} = h \cot\phi \tag{6}$$

We use (6) to show mapping of $P_3 = [x_3, y_3, z_3, 1]$ and $P_4 = [x_4, y_4, z_4, 1]$ from world coordinates to image coordinates $p_3 = [su_3, sv_3, s]$ and $[su_4, sv_4, s]$ respectively.

$$\begin{bmatrix} su_3 \\ sv_3 \\ s \end{bmatrix} = P \begin{bmatrix} x_3 \\ h\cot\phi \\ 0 \\ 1 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} su_4 \\ sv_4 \\ s \end{bmatrix} = P \begin{bmatrix} x_4 \\ h\cot\phi \\ 0 \\ 1 \end{bmatrix} \tag{8}$$

Expanding and rearranging the above expressions to solve for camera height:

$$h = \frac{f(x_4 - x_3)\sin\phi}{u_4 - u_3} \tag{9}$$

Since we know the average width *w* of a vehicle, we substitute

$$x_4 - x_3 = \frac{w}{\cos\theta}$$

in the previous equation to arrive at:

$$h = \frac{wf\sin\phi}{(u_4 - u_3)\cos\theta}$$

Finally, to compute speed of a vehicle we need to know the distance traveled by that vehicle. Road coordinates can be easily obtained from image coordinates by substituting *z = 0* in (3) and solving for *x* and *y*.

$$x = \frac{uh}{v\cos\phi + f\sin\phi} \qquad y = \frac{h(f - v\tan\phi)}{v + f\tan\phi}$$

References

1. Neeraj K. Kanhere, Stanley T. Birchfield and Wayne A. Sarasua. Vehicle Segmentation and Tracking in the Presence of Occlusions. In *TRB Annual Meeting Compendium of Papers, Transportation Research Board Annual Meeting*, January 2006.

2. Neeraj K. Kanhere, Shrinivas J. Pundlik and Stanley T. Birchfield. Vehicle Segmentation and Tracking from a Low-Angle Off-Axis Camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1152-1157, June 2005.

3. S. Gupte, O. Masoud, R. F. K. Martin and N. P. Papanikolopoulos. Detection and Classification of vehicles. In *IEEE Transactions on Intelligent Transportation Systems*, Vol. 3 (1), pages 37-47, March 2002.

4. D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real time computer vision system for measuring traffic parameters. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 495–501, 1997.

5. Miroslav Trajkovic. Interactive Calibration of a PTZ Camera for Surveillance Applications. In *Asian Conference on Computer Vision*, 2002

6. E. K. Bas and J. D. Crisman. An easy to install camera calibration for traffic monitoring. In *Proceesings of the IEEE Conference on Intelligent Transportation Systems*, pages 362-366, 1997.

7. H. S. Lai. Vehicle extraction and modeling, an effective methodology for visual traffic surveillance. PhD Thesis, Chapter 5, The University of Hong Kong, 2000

8. George S.K. Fung, Nelson H. C. Yung and Grantham K. H. Pang. Camera calibration from road lane markings. In *Optical Engineering*, vol. 42, pages 2967-2977, October 2003.

9. Xiao Chen He and Nelson H. C. Yung. New method for overcoming ill-conditioning in vanishing-point-based camera calibration. In *Optical Engineering*, Vol. 46 (3), 2007.

10. D. Dailey, F. W. Cathy and S. Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. In *Proceesings of the IEEE Conference on Intelligent Transportation Systems*, pages 98-107, 2000.

11. Todd N. Schoepflin and Daniel J. Dailey. Dynamic Camera Calibration of Roadside Traffic Management Cameras for Vehicle Speed Estimation. In *IEEE Transactions on Intelligent Transportation Systems*, Vol. 4(2), pages 90-98, June 2003.

12. Kai-Tai Song and Jen-Chao Tai. Dynamic Calibration of Pan-Tilt-Zoom Cameras for Traffic Monitoring. In *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 36(5), October 2006.

13. Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Comunications of the ACM*. Vol. 24(6), pages 381-395, 1981.

14. New Jersey department of transportation, Roadway Design Manual.
http://www.state.nj.us/transportation/eng/documents/RDME/sect2E2001.shtm
Accessed on March 17, 2007.

15. Viola P. and Jones M. Rapid object detections using a boosted cascade of simple features. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1, pages 511-518, 2001.

16. Chee-Woo Kang, Rae-Hong Park and Kwae-Hi Lee. Extraction of straight line segments using rotation transformation: generalized Hough transformation. Pattern Recognition, Vol 24 (7), pages 633-641, 1991.

17. Hartley, R. I. and Zisserman, A. Multiple View Geometry in Computer Vision (second edition), 2004. Cambridge University Press, ISBN: 0521540518.

18. Intel OpenCV library. http://www.intel.com/technology/computing/opencv/index.htm
Accessed on March 12, 2007.