Real-time Vehicle Detection for Highway Driving

Ben Southall, Mayank Bansal and Jayan Eledath
Sarnoff Corporation
Princeton, NJ
{bsouthall,mbansal,jeledath}@sarnoff.com

Abstract

We present a new multi-stage algorithm for car and truck detection from a moving vehicle. The algorithm performs a search for pertinent features in three dimensions, guided by a ground plane and lane boundary estimation sub-system, and assembles these features into vehicle hypotheses. A number of classifiers are applied to the hypotheses in order to remove false detections. Quantitative analysis on real-world test data show a detection rate of 99.4% and a false positive rate of 1.77%; a result that compares favourably with other systems in the literature.

1. Introduction

This paper presents a stereo vision based algorithm for vehicle detection from a moving car. In particular, our algorithm is focused on short-to-medium range (3 to 40 meter) vehicle detection and tracking for the purpose of low speed automated cruise control (ACC) and stop and go vehicle control in moderate to heavy traffic. There are many algorithms in the literature for vehicle detection and tracking from a moving platform; a full treatment of the background can be found in a recent survey by Sun *et al*. [16]. Our system follows the typical design described by Sun and co-authors, with a hypothesis generation step where the algorithm locates regions of interest (ROI) in an image, followed by a hypothesis verification step, where these ROI are subject to further analysis to determine whether vehicles are present or not. The main novelty of our system is in the combination of its particular component algorithms; this combination permits us not only to produce classification outputs whose accuracy is greater than other systems from the literature, but also to give range estimates to vehicles ahead of the host car, with all of these results delivered at a frame-rate of 16Hz.

In both monocular and stereo vision systems, image edges are popular cues for hypothesis generation [15, 7, 13, 1, 8], with authors such as Bergmiller *et al*. [2], and Bertozzi *et al*. [3] adding extra features such as symmetry or shadow presence. Broggi *et al*. [4] use edge and symmetry features in a multi-resolution framework. Other authors [14, 9] scan the image with lightweight (or hardware optimized) SVM classifiers to locate likely vehicle locations.

Franke and Kutzbach [8], use a set of filters to generate a sparse set of edge points, estimate the disparity of those points, and build a histogram of those disparities. The vehicle detection proceeds on the principle that the rear faces of vehicles are quite flat, and therefore should have constant depth, and hence constant disparity. Given this fact, the algorithm detects likely car regions by locating peaks in the disparity histogram, finding the image features that contributed to the peak, and then fitting rectangular ROI around some subset of those features. Knoeppel *et al*. [10] use the same technique, but with a different low-level feature extraction procedure.

Hypothesis verification is a classification step, where hypotheses are sorted into vehicles and non-vehicles. Richter *et al*. [12] attempt to form 'U' shaped contours that would describe the sides and bottom edge of a car in image regions cued by a radar detector; the validity of detections is determined by properties of the fitted contour. Song *et al*. [13] use an AdaBoost classifier [17] trained on Haar features to classify detections. Broggi *et al*. [4] eliminate bounding boxes that have either dimensions or locations in the image that are unlikely to describe vehicles. Alonso *et al*. [1] use a trained nearest neighbour approach to separate vehicles from non-vehicles based upon features such as image corners, symmetry and shadow presence. Song *et al*. [15] test both neural network and support vector machine classifiers with Gabor and wavelet features as inputs. Franke and Kutzbach make use of a ground plane detection scheme to remove those hypotheses that largely contain image features that lie on the road surface.

There are quantitative results for a number of the algorithms that use trained classifiers in the hypothesis verification stage, and a summary of those results are presented in table 1 alongside ours, which compare favourably. Much of the remaining literature in this area concentrates on description of algorithms, with algorithmic performance being

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

Authors	TP rate	FP rate	Notes
Bergmiller *et al*. [2]	79.84%/83.12%	16.7%	TP rates for cars/trucks
Sun *et al*. [15]	98.5%	2%	For their most effective classifier
Alonso *et al*. [1]	92.63%	3.68%	
Fu *et al*. [9]	87.6%	-	No FP rate given. TP rate an average over 10 sequences
Our results	**99.37%**	**1.77%**	Also delivers vehicle range, 16Hz frame rate

Table 1. Exisiting results

assessed in a qualitative manner.

Our approach shares some broad principles with many of the works discussed above, although our implementations differ. For hypothesis generation we employ a stereo-based detection scheme that exploits regions of constant disparity [8], and we also use multi-resolution image processing [4]. As we will see below, the novelty of our hypothesis generation algorithm is in a multi-resolution search constrained by so-called 'detection gates' that are guided by lane boundary location and ground plane estimates generated by an auxiliary algorithm. Parts of our hypothesis verification scheme utilize a trained classifier, bolstered by a set of simple pre-classifiers that exploit expected features of our operating environment.

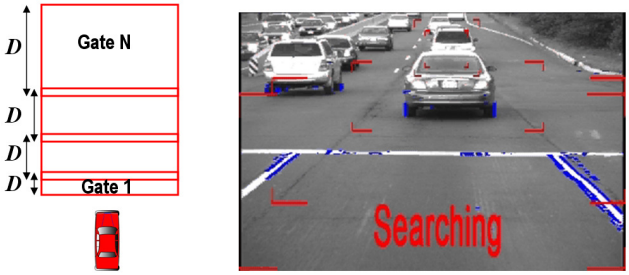

Figure 1. Detection gates. Left: An overhead view showing a set of N gates ahead of the vehicle, each of which covers an equal number D stereo disparities. Right: Red reticles show the detection gates projected onto the image plane. Blue pixels show ground plane points.

2. Approach

Given that the purpose of our detection and tracking system is to provide data to a real-time control system, we designed the algorithms with two requirements in mind. The first is the real-time constraint; the algorithms must have potential to run at frame rates of 10Hz or more on low-cost automotive grade embedded computing hardware. The second requirement is to minimize error rates. The driver expects an ACC system to react to the presence of vehicles in the lane ahead; if the system misses vehicles, then it does not provide any value to the driver and they will switch it off. Conversely, frequent false positives (i.e. it falsely reports a vehicle where non is present) will cause the car to brake or accelerate in a disconcerting manner. In the worst case, either of these errors could cause a safety hazard.

To satisfy the timing constraints, we designed the algorithms such that the more computationally complex procedures only occur on relatively small amounts of data, and where we must process large amounts of data (e.g. in low-level image processing for feature extraction), we use simple operations. To provide low error rates, we design our processing chain such that at the lowest level we are unlikely to miss necessary features for vehicle detection (and thereby lower the risk of false negatives), and in later processing we use multiple simple cues to reduce false detections.

The basic assumption of our stereo-based vehicle detection strategy is that of Franke and Kutzbach [8], where they note that image features on the rear face of a vehicle ahead of the host will share a similar depth, and hence a similar disparity value. Accordingly, our basic detection algorithm searches for sets of image features that are clustered in disparity space. To speed up computation and reduce the likelihood of errors, we focus our search for vehicles on a set of so-called 'detection gates'; each detection gate defines a 3D volume on the roadway ahead of the vehicle, and individual instances of our detection algorithm run separately in each gate. A set of N detection gates are illustrated in the overhead view on the left-hand side of figure 1. The width of each gate is that of a typical US highway lane ($3.6m$), and they are 1.8 m tall. The length of the each gate is specified to equal a disparity of D pixels. Equal length in disparity space implies that the gates lengthen as they get further from the host car; additionally, the gates overlap in the longitudinal direction (by 1 pixel of disparity).

In order to focus our search for vehicles on the surface of the highway lane ahead of the host, we must also steer these gates in response to two changes; pitching motion of the host as it drives along the road, and the location of the lane boundaries as the road curves, or as the host changes lanes. To this end, our system incorporates a sparse stereo ground plane location algorithm (based upon similar principles as the work presented by Labayrade *et al*. [11]), coupled with a lane boundary extraction procedure. On the right-hand side of figure 1 we see an image from our system with some graphical overlays. The red corner reticles show the projection of the front of each 3D detection gate onto the image plane. The blue point features indicate those sparse points that have been identified as lying on the ground plane; as

the vehicle pitches, the ground plane detection algorithm allows us to move the detection gates up and down to match the motion of the imaged ground plane. The 3D ground plane points are further analyzed to extract lane boundary poly-lines that in turn are used to steer the detection gates laterally in the image in sympathy with the curvature of the lane ahead of the host.

Finally, there is one more focus mechanism that we use to increase the effectiveness of our processing; image scale. Our detection algorithm searches for image features that describe the outer edges and significant inner structure of the rear face of a vehicle. These features have a constant physical size in the world, but naturally change size in the image as the observed vehicle moves relative to the camera. In order to keep these physical features within a small range of image scales, we use a Laplacian pyramid decomposition of our original image, with nearer gates using lower resolution parts of the pyramid, and the most distant gates using the original image resolution. This strategy allows us to use a single set of filters for extracting image features for vehicles at all ranges; this choice will ease the transition to DSP hardware that is optimized for single instruction multiple data parallelism. A further feature of the multiple-resolution algorithm is that it helps to preserve the 'constant disparity' assumption for rear faces of vehicles over a greater range. As the disparity resolution effectively drops with image resolution for nearby vehicles, the amount of disparity variation caused by rotations of these vehicles also drops. For typical U.S. highways (curves of $> 350m$ radius), we have found that the constant disparity assumption is well preserved across the operating range of our system.

Once a vehicle has been found in one of the detection gates, a tracker is initialized to estimate the trajectory of the vehicle ahead; for our ACC application, we track only the nearest vehicle in the lane of the host vehicle. A track is terminated if the tracked vehicle moves out of the host's lane (or if the host performs a lane change), if the lead vehicle moves out of the reliable working range of the system, or if a new vehicle cuts in to the host's lane at a distance closer than the currently tracked vehicle (in which case the cutting-in vehicle is tracked).

To summarize, our stereo-based vehicle detection scheme operates multiple versions of the same basic algorithm in a carefully guided set of physical neighbourhoods, with input from a ground plane and lane boundary determination algorithm. We use a multi-scale approach to image processing such that the image features that serve as input to our algorithm are likely to correspond to the same physical features on a vehicle across the full operating range of our system. In subsequent sections, we provide more details of our algorithm's components, our experimental procedure and the results of those experiments.

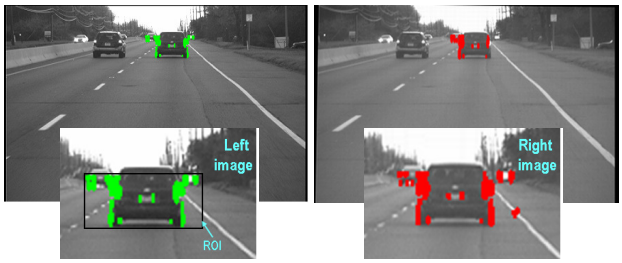

Figure 2. Edge extraction. The left and right images are shown, together with their detected edge pixels in green and red for the left and right images respectively. The detail images give a closer view, and the left image detail includes a detection gate ROI.

3. Algorithmic details

3.1. Vehicle detection

As noted above, our system runs several independent instances of our vehicle detector on different parts of the road ahead of the host vehicle. Below, we provide more details on how the vehicle detection algorithm is structured. We first address feature selection and range computation, then vehicle candidate detection in a single detection gate, merging results from multiple detection gates, and finally outlier rejection schemes to reduce false alarms.

3.1.1 Feature generation and range computation

Our detection algorithm attempts to locate regions of constant disparity in each of the detection gates. To this end, we generate vertical edge pixels using a standard vertical Sobel edge filter, cluster those pixels into vertical edgels and then compute the range to each edgel using standard stereo SAD search techniques. As seen in the algorithm overview section, our algorithm runs at multiple resolutions, with nearby detection gates operating at coarse resolution, and the furthest gates operating at the original image resolution. Each gate covers a region of interest (ROI). To reduce computation, at each pyramid level we take a union of the ROIs for the detection gates that operate on that level and only compute edges and stereo inside the union ROI. To compensate for varying light levels, we use an adaptive thresholding scheme to determine which pixels belong to significant edges. Algorithms 1 and 2 show the steps of the feature detection and ranging process, and fig. 2 shows an example of the edge extraction process.

3.1.2 Per-gate candidate selection

Now that we have a set of vertical clusters c at each image resolution, with known image column location $x(c)$ and disparity $D(c)$, we construct a data structure that is used to generate input for our vehicle detection algorithm. The data structure is a straightforward 2D histogram where columns

Algorithm 1 The multi-resolution feature extraction algorithm

for all Left image pyramid levels p **do**
  Select level-appropriate ROI
  Apply vertical Sobel filter across ROI
  Choose threshold to select top 15% of non-zero edge pixels
  Binarize left edge image according to this threshold
  Store the number $N(p)$ of above threshold pixels
end for
for all Right image pyramid levels p **do**
  Select level-appropriate ROI
  Apply vertical Sobel filter across ROI
  Compute a threshold that maintains $N(p)$ edge pixels for the right edge image
  Binarize right edge image according to this threshold
end for

Algorithm 2 Depth estimation and x-D histogram construction

for all Left edge image pyramid levels p **do**
  Form vertical clusters of edge pixels c_p via run-length encoding over each column x
  for all Vertical edge clusters c_p **do**
    Store column location $x(c_p)$, and number of pixels $N(c_p)$
    Perform SAD search to find best disparity match $D(c_p)$ on the right edge image
  end for
end for

of the histogram correspond to columns of the image, and the rows of the histogram correspond to increments of disparity. We populate one of these so-called x-D Histogram for each pyramid level by adding $N(c_p)$ (the number of pixels in a cluster) votes for each vertical cluster feature found at that level. For each populated x-D histogram, we then extract the local maxima using the mean shift algorithm [5], as shown in algorithm 3. Fig. 4 shows the x-D histogram generated for a single vehicle.

Algorithm 3 x-D Histogram construction

for all Pyramid levels p **do**
  for all Vertical edge clusters c_p **do**
    Add $N(c_p)$ votes to the bin $x(c_p), D(c_p)$ of x-D Histogram p
  end for
  Perform mean-shift on the x-D Histogram p to extract the maxima M_p
end for

The vehicle detection algorithm analyses, on a per-gate basis, the maxima extracted from the x-D histogram. The algorithm first computes a coarse disparity D_g for each gate g; D_g is the disparity value that maximises the overlap of the edge points that lie within the gate's image ROI in the left and right edge pyramid; fig. 3 shows an image overlay where the edge points from the right image (in red) have been shifted by D_g pixels so that they best overlap the edge pixels from the left image (in green). D_g permits a coarse estimation of depth to the dominant features within a gate, and permits us to compute the expected image width w_x, in pixels, of a vehicle at that depth if one is present. This nominal width w_x is used to select pairings of peaks in the x-D histogram whose separation is close to the width of a vehicle. The pair of peaks that satisfy the width requirement and have the largest combined support are chosen as the outer edges of a detected vehicle, and an image ROI is derived that encompasses all edge points that contribute to the chosen peak pair, and any peaks that lie between them in the x-D histogram. Fig. 5 shows the final output of this algorithm. If no suitable peak pairing can be selected, the algorithm returns no detection.

Figure 3. Coarse edge point alignment.

3.1.3 False positive rejection

Not all peak pairs chosen by our algorithm correspond to vehicles; road markings and other scene structures can sometimes generate 3D peak pairings that generate false detections from our algorithm. To reduce these false positives, we run four successive false positive rejection schemes. The first is based upon computing a consensus score between the detection ROI as projected onto the left and right edge pyramids. It two selected edges span a region that does contain a flat vehicle-like surface, then the edge pixels between the peaks will all be well explained by the refined disparity value and the consensus score will be high; if the edges do not encompass a vehicle, then the edge pixels between the peaks will not align well under the refined dis-

Algorithm 4 Vehicle detection

for all Detection gates g **do**
Select gate pyramid level p and gate ROI
Estimate coarse disparity D_g that maximises overlap between edge pixels in left and right edge pyramid in the ROI set by gate g
Use D_g to set the expected with w_x, in pixels, of a vehicle-sized object
Choose from M_p the set of peaks, M_g, that lie inside g
Choose from M_g all pairs of peaks whose separation in x is close to w_x
Choose the pair of peaks with the strongest combined edge support
Detected vehicle ROI encompasses all edge pixels that belong to peaks in M_g that lie between the two selected peaks in the x-D histogram
Detected vehicle refined disparity D_{gref} is computed as the mean of the disparity of edge pixels in the ROI
end for

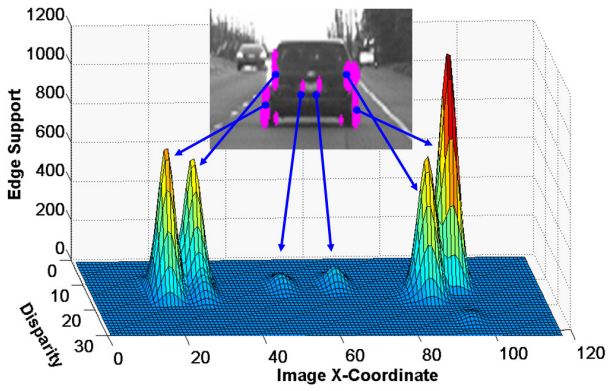

Figure 4. x-D Histogram. The extracted edge clusters, shown in purple, are projected into the x-D histogram. Blue lines denote the connection between image features and the corresponding peaks in the histogram.

Figure 5. The detection resulting from our x-D histogram analysis

parity, and consensus will be low. The second rejection method analyses the horizontal edge structure within the detection ROI; vehicles typically contain at least one of multiple strong horizontal features such as the top and bottom edges of the vehicle, the fender, rear windshield surrounds and so on. The third rejection scheme compares the edge pixels selected by the detection ROI with those edge pixels that are classified as belonging to the ground plane (e.g. the blue pixels on the right of fig. 1); true detections will not contain many pixels that lie on the ground. The final rejection scheme runs a simple AdaBoost classifier [17], using histogram of oriented gradients (HOG) features [6], on the detection ROIs that have survived the previous stages of pruning. More details can be found in algorithms 5 - 8. The thresholds found in those algorithms were selected after experimentation with a range of different data sets. In a cascaded false positive rejection scheme as the one described here, there is no mechanism to recover from early rejection of positive cases, so the goal during threshold selection was to maximise the number of true positives, and to rely upon the next stage to further reduce false positives.

Algorithm 5 Disparity consensus based false positive rejection

for all Detection ROIs **do**
Choose the pyramid level p for the detection
Count N, the number of edge pixels inside the detection ROI
Count N_c, the number of pixels whose location in the right image is consistent with those in the left edge image and refined disparity D_{gref}
if $N_c/N > 0.7$ **then**
Accept the detection, and store the value of N_c/N
else
Reject the detection
end if
end for

Algorithm 6 Horizontal structure based false positive rejection

for all Detection ROIs **do**
Store width W of the ROI
Convolve left image with a horizontal Sobel filter
Compute horizontal edge clusters
Find length L of longest edge cluster
if $L/W > 0.8$ **then**
Accept the detection
else
Reject the detection
end if
end for

3.1.4 Multiple-detection elimination

As shown in figure 1 (left), there is overlap between the detection gates , and if a vehicle is present in the overlap

Algorithm 7 Ground plane based false positive rejection

for all Detection ROIs **do**
  Retrieve vertical edge pixels and count their number N
  Retrieve ground plane points
  Count $N_{overlap}$, the number of ground plane pixels that overlap detection ROI edge pixels
  if $N_{overlap}/N < 0.35$ **then**
    Accept the detection
  else
    Reject the detection
  end if
end for

Algorithm 8 AdaBoost false positive rejection

for all Remaining detection ROIs **do**
  Map detection ROI left image data onto 128×64 grid
  Compute histogram of gradients features on rescaled image data
  Compute classification score s for HOG features
  if $s > 0$ **then**
    Accept the detection
  else
    Reject the detection
  end if
end for

region, it can be detected in more than one gate. We identify and eliminate multiple detections of a single vehicle using algorithm 9, which makes use of the consensus test score N_c/N (computed in algorithm 5).

Algorithm 9 Multiple detection removal

Form a graph G where vertices are detections
for all Detection d_i **do**
  for all Detection $d_j, i \neq j$ **do**
    if ROIs for d_i and d_j overlap by 80% AND disparities for d_i and d_j are within 1 pixel **then**
      Join d_i and d_j with an edge
    end if
  end for
end for
Find the connected cliques C of G each clique represents a (multiple) detection
for all Cliques C **do**
  Find the detection with the largest consensus score N_c/N
  Keep this detection and delete the other detections in the clique
end for

3.1.5 Temporal consistency and tracker initialization

After first detection, a candidate becomes a tentative track and we start a simple temporal consistency test before confirming the track. If a candidate is present for 4 out of 5 frames in sequence, we declare the tentative track confirmed. If there is no vehicle currently being tracked, we now initialize the tracker with the confirmed detection. If there is a vehicle being tracked, and the newly confirmed track is closer to the host than the tracked vehicle, then the tracker will be terminated and the newly confirmed candidate is used to initialize a new track. If the candidate is more distant from the host vehicle than the current track, then the new candidate is not tracked.

3.2. Vehicle tracking

We use an extended Kalman filter to estimate the position, velocity and acceleration of the lead vehicle, and also its width; the state evolution model is a simple linear constant acceleration model, and the measurement model is non-linear to reflect the transformation of our observations from image and disparity co-ordinates into location and depth in the world. The most important property to note here is that the tracker uses its own detection gate that is guided by the Kalman filter; the tracker's gate is permitted to move longitudinally with respect to the vehicle, and through image pyramid levels as the tracked vehicle moves toward and away from the host. We track a single target until either the target moves out of the host vehicle's lane (or vice versa), a decision aided by our lane boundary estimator, or until the target moves beyond the maximum range of our system.

4. Experiments

4.1. Procedure

Our experimental goal is to measure the detection accuracy of our system; in particular the true positive and false positive rates are of interest. In the experiments presented here, we collected 44 image sequences, with a total of more than 31,000 frames, that cover a range of realistic on-road operating conditions and vehicle types, including both cars and trucks. To generate ground truth, we developed a semi-automatic annotation tool that uses a mixture of human-guided track initialization and correction, combined with dense stereo data (generated offline on the left and right input images using a region-based SAD algorithm) and automatic tracking. The user loads a data sequence, and steps through the data until first frame where a vehicle is within range of the system and in the lane ahead of the host. At this point, the user clicks on the rear face of the vehicle, and the dense range data for that part of the image is used in conjunction with our edge finding routines to generate a ROI

that is likely to enclose the vehicle. If this ROI is not accurate, the user can correct it manually by drawing with the mouse. Once an accurate initial ROI is generated, the user lets the system run with the same automatic tracker used in the experimental system; if the tracker produces inaccurate ROIs at any frame, the user can rewind to those frames and manually correct the ROI so that it properly surrounds the vehicle. If the lead vehicle moves out of the host vehicle's lane, the user manually terminates the track, and reinitializes at the next point a vehicle is in the lane ahead of the host.

The ground truth files generated by our tool record the location of the ROI and average dense stereo range to the lead vehicle for each frame that such a vehicle is present. During testing, we save to file the output ROI and estimated range (calculated from the refined vehicle disparity) for each frame that our vehicle tracker is active. We consider the ground truth and tracker output to be in agreement for a given frame if the tracked ROI is contained within a bounding box 20% wider than the ground truth ROI. If the tracked output ROI does not satisfy this constraint, we have a false negative. If the tracker data shows an output for a given frame where the ground truth does not, we have a false positive.

In the results below, data was collected from a stereo sensor with NTSC cameras with lenses of 48.2 degrees field of view, and a 7" baseline separates the two cameras. The maximum working range of this system is 40 meters. We also have a long range system with the same cameras and baseline but 15 degree field of view lenses. This system operates out to 100 meters with similar results.

4.2. Results and Discussion

Table 2 shows the results of running our algorithm on data sets collected on the road for cars and trucks separately, with 'total' showing the overall combined result. We present both a true positive and false positive rate, and also the average number of frames between false positives. Our true positive and false positive rates compare favourably against those presented in table 1, although a direct comparison is not possible given that the data sets used in all experiments differ.

Fig. 6 shows the detection rates (bottom graphs) and the number of false positives per frame (upper graphs) on a per-sequence basis. The mean of the quantities is plotted in purple. It can be seen that a few sequences distort the averages. In the case of trucks, sequence 7 shows a higher false positives per frame rate, and a lower detection rate. Both of these were due to the same failure; the tracker became distracted for 36 frames, causing at once both the false positives (tracking the distraction) and false negatives (missing the truck that was present). In the case of the car data, sequence 29 shows a similar failure. If we neglect these two

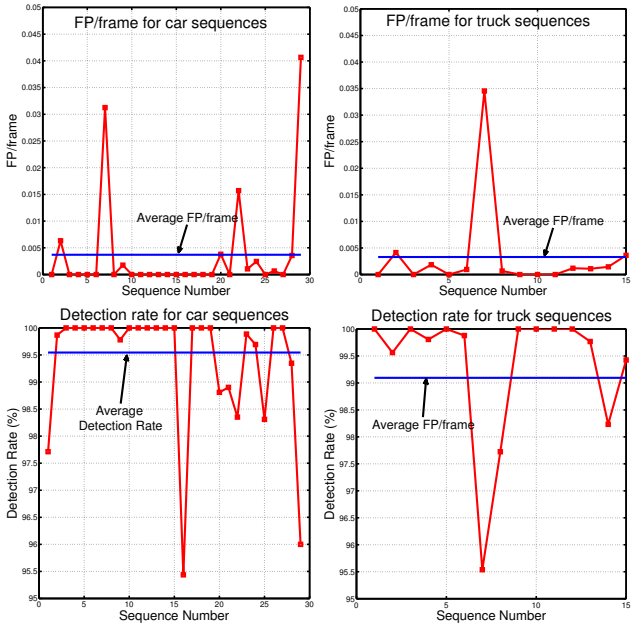

Figure 6. Detection and false positive rates.

sequences, our overall detection rate rises to 99.5%, and our average number of false positives per frame drops to 0.002, or one false positive every 500 frames. We note that the plot of false positives per frame indicates that false positives tend to be episodic, and confined to a few sequences where our AdaBoost classifier is ineffective rather than being a general problem everywhere. Given the tiered false-positive rejection that we designed into our system, this type of error is preferable; as more sophisticated outlier rejection schemes become available, they can be naturally incorporated into our architecture to improve performance.

We also present some visual output from our long-range narrow field of view system in figure 7.

5. Discussion and Conclusion

We have described an algorithm for the detection of vehicles in the lane ahead of a host vehicle. Off-line detection results have been presented and are shown to be superior to other published results both in terms of true positive and false positive rates. There is scope to drive the number of false positives per frame down to provide the kind of reliability that an automotive control system requires, and this shall be a certain focus of future work. The results suggest that false detection errors are episodic in nature rather than pervasive, and that these episodes are somewhat rare, which suggests that fixing specific errors will lead to discrete improvements in performance.

The algorithm has been designed to run on relatively lightweight computing hardware; the current online version runs, with live graphical display, at a rate of 16Hz on a Pen-

	# Seq.	# Frames	TP rate	FP rate	Mean FP/frame
Car	29	18903	99.55%	1.86%	0.00365
Truck	15	12461	99.09%	1.66%	0.00409
Total	44	31364	99.37%	1.77%	0.00383

Table 2. System performance. TP = true positive, FP = false positive.

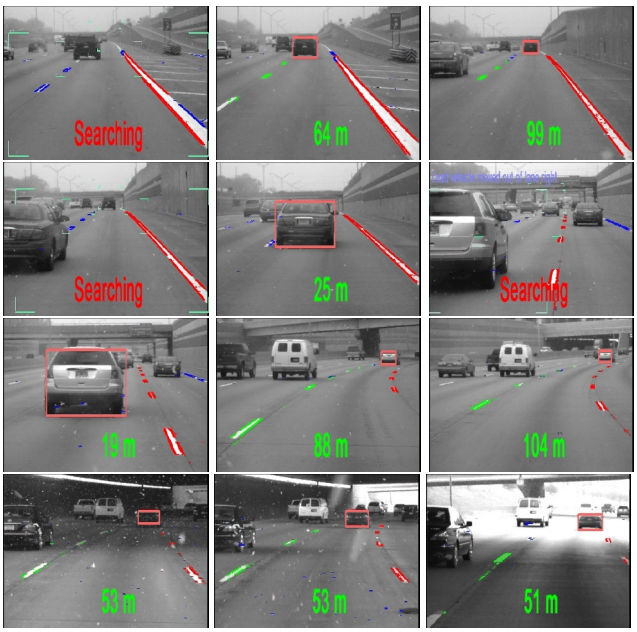

Figure 7. Graphical output of the long range system, including vehicle cut in, host lane change and going underneath a wide overpass.

tium 4 processor. Many parts of the algorithm are candidates to be run in parallel, and ripe for DSP optimization en route to automotive hardware implementation, a goal for future work.

Acknowledgements

We would like to thank Autoliv Electronics America for financial support and data.

References

[1] D. Alonso, L. Salgado, and M. Nieto. Robust vehicle detection through multidimensional classification for on board video based systems. *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, 4:IV –321–IV –324, 16 2007-Oct. 19 2007.

[2] P. Bergmiller, M. Botsch, J. Speth, and U. Hofmann. Vehicle rear detection in images with generalized radial-basis-function classifiers. *Intelligent Vehicles Symposium, 2008 IEEE*, pages 226–233, June 2008.

[3] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele. Stereo vision-based vehicle detection. In *in IEEE Intelligent Vehicles Symposium*, pages 39–44, 2000.

[4] A. Broggi, P. Cerri, and P. C. Antonello. Multi-resolution vehicle detection using artificial vision. In *IEEE Intelligent Vehicles Symposium*, pages 310–314, 2004.

[5] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, May 2002.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1:886–893 vol. 1, June 2005.

[7] F. Dellaert and C. Thorpe. Robust car tracking using kalman filtering and bayesian templates. In *Conference on Intelligent Transportation Systems*, 1997.

[8] U. Franke and I. Kutzbach. Fast stereo based object detection for stop & go traffic. *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, pages 339–344, Sep 1996.

[9] C.-M. Fu, C.-L. Huang, and Y.-S. Chen. Vision-based preceding vehicle detection and tracking. *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2:1070–1073, 2006.

[10] C. Knoeppel, A. Schanz, and B. Michaelis. Robust vehicle detection at large distance using low resolution cameras. *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 267–272, 2000.

[11] R. Labayrade, D. Aubert, and J.-P. Tarel. Real time obstacle detection on non flat road geometry through 'v-disparity' representation. In *Proceedings of IEEE Intelligent Vehicle Symposium*, volume 2, pages 646–651, Versailles, France, 2002. http://perso.lcpc.fr/tarel.jean-philippe/iv02.html.

[12] E. Richter, R. Schubert, and G. Wanielik. Radar and vision based data fusion - advanced filtering techniques for a multi object vehicle tracking system. *Intelligent Vehicles Symposium, 2008 IEEE*, pages 120–125, June 2008.

[13] G. Y. Song, K. Y. Lee, and J. W. Lee. Vehicle detection by edge-based candidate generation and appearance-based classification. *Intelligent Vehicles Symposium, 2008 IEEE*, pages 428–433, June 2008.

[14] G. Stein, E. Rushinek, G. Hayun, and A. Shashua. A computer vision system on a chip:a case study from the automotive domain. In *EmbedCV05*, pages III: 130–130, 2005.

[15] Z. Sun, G. Bebis, and R. Miller. Monocular precrash vehicle detection: Features and classifiers. *IEEE Transactions on Image Processing*, 15(7):2019–2034, 2006.

[16] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:694–711, 2006.

[17] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.