Realtime On-Road Vehicle Detection with Optical Flows and Haar-Like Feature Detectors

Jaesik Choi
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

An autonomous vehicle is a demanding application for our daily life. Such vehicle requires to detect other vehicles on the road. Given the sequences of images, the algorithms need to find other vehicles in realtime. There two types of on-road vehicles, traveling in the same direction or traveling in the opposite direction. Due to the distinct features of two types of vehicles, different approaches are necessary to detect vehicles in different directions. Here, we use 'optical flow' to detect vehicles in the opposite direction because the coming traffics shows salient motions. We use 'Haar-like feature detection' for vehicles in the same direction because the traffics represent relatively stable shape (car rear) and little motion. We verify the detected region with estimating 3D geometry. If the detector fail to find the vehicles, we interpolate the region from the previous frames. Then, the detected vehicles are projected into the 3D space. Our system detects vehicles with high accuracy in realtime for 11 frames/sec.

1. Introduction

A vehicle detection algorithm is one of the most important modules of an autonomous vehicle system because it detects midrange and long distance (over 20m) vehicles with low cost. However, any vehicle detection algorithm can not detect all vehicles. Here, we consider detecting two types of cars. The two types of cars (traveling in the opposite direction and in the same direction) have distinct features.

We use 'Optical flow' [5] to detect coming traffics. Given the two consecutive frames, we find corner points with corner detector [7]. The detected features are matched by the 'Optical flow' algorithm [5]. The 'Optical flow' assumes that important features are detected in both frames. We cluster some features into a rectangle if the euclidian distance of two features (the location of feature and the direction of optical flow) are small. The optical flow algorithms find the correspondence within a reasonable time so that we can use the algorithms for realtime application [1]. However, some of optical flows are generated by some cracks of the road and road signs. They make false positive rectangles.

Haar-like feature detector [11] is used to detect traffics in the same direction. The optical flow algorithm is not so appropriate for these traffics because they do not have any salient movement in most case. Instead, they maintain some consistent shapes. The shapes mostly show the rear side of the car. To detect the rear shape of the car we choose to use Haar-like feature detector because it is fast and efficient. In general, template-based methods are slow for realtime detections. The execution time of the methods is proportional to the number of templates we have. That is, the execution time increases when we have more templates. However, Haar-like feature detector searches the frame once. The detector also generates many false positives because the detector represents the various shapes of the vehicles. If we only focus on the shapes of the cars in the training data set, the detector easily finds those cars. However, the detector may fail to find the cars which are not trained. It is dangerous for the autonomous vehicle. Thus, we loosen the constraints of the detector and receive some false positive detections.

Many false positive rectangles (or detections) can be filtered out, when we process the image based on the context of the scene [4]. We estimate 3D geometry and use it to impose the constraints of geometry. If we find a vanishing point in a frame, the rectangle (hypothesis) can be verified by its size. We assume that the sizes of cars are in certain ranges. If the cars locate near the vanishing point, the sizes should be small. If the cars are far from the vanishing point, the sizes should not so small. Of course, any car can not be

[1] real-time is a different term with the term in real-time operating system. The real-time in the operating system means that the system always finish its works within appointed time. Here, the real-time in some papers means that the processing time of vehicle tracking is faster than the frame rate. That is, there is no guarantee to meet the dead line

on the sky. With this simple method, we filtered out may false positive detections.

Although there are many researches for realtime on-road vehicle detection [1, 9, 13] so far, there is no research that use Haar-like feature detector, optical flow and scene context simultaneously on the moving vehicle with monocular camera. Recently some researchers [8], [12] have improved the performance of Viola's detector. Miolos [8] reduced the time for training parameters and Jianyu [12] focused on improving the tracking performance with particle filtering. However, those are not designed for real-time detection of on-road vehicles.

In general, a vehicle detection algorithm has two basic step; Hypothesis Generation (HG) and Hypothesis Verification (HV) [6]. In the Hypothesis Generation (HG) step, the algorithm hypothesizes the locations of vehicles in an image. In the Hypothesis Verification (HV) step, the algorithm verify the presence of vehicles in an image. The methods in the HG step can be categorized into three methods; Knowledge-based methods using symmetry of object, color, corners and edges; Stereo-Vision-Based methods using two cameras; Motion-Based Methods tracking the motion of pixels between the consecutive frames (optical flow[5]). The methods in the HV step are Template-based methods and Appearance methods. Template-based methods use predefined patterns of the vehicle class. Appearance-based methods include pattern classification system between vehicle and non-vehicle.

Three previous works [1, 9, 13] tried to solve realtime on-road vehicle detection problem. All the papers used monocular cameras and have real-time constraints. [1] used horizontal and vertical edges (Knowledge-based methods) in HG step. The selected regions at HG step are matched with predefined templates in HV step. [9] used horizontal and vertical edge in HG step. However, they use Haar Wavelet Transform and SVMs (Appearance-based methods) in HV step. [13] detected long-distance stationary obstacles including vehicles. They used an efficient optical flow algorithm [3] (Motion-based methods) in HG step. Although they did not use any traditional HV method, they used Sum of squared differences(SSD) with a threshold value to verify their hypothesis.

2. Theories

This works use three prevail methods in computer vision. They are 'optical flow' [5], 'Haar-like feature detector' [11] (so-called Viola and Jones), and 'Scene analysis with context' [4]. A normalized input image is simultaneously processed by 'optical flow' and 'Viola and Jones detector'. The hypothesis are analyzed by 'Scene analysis with context'.

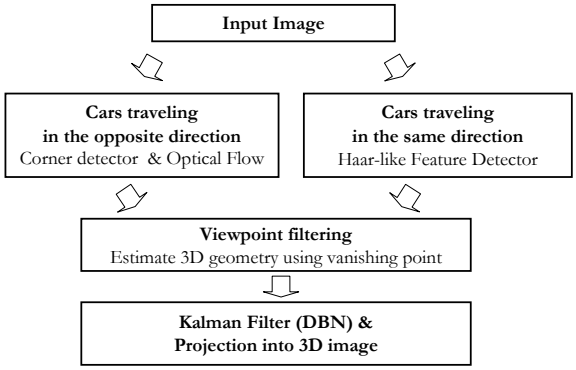

Figure 1: Overall diagram of on-road vehicle detection algorithms.

2.1. Optical Flow

Overall process of optical is following: First, the algorithm finds the important features (corners) in the two consecutive frames; Second, the algorithm matches the correspondence between the features (corners); Finally, extracted flows are clustered into a group if the Euclidean distances of the flows (location and direction) are small.

2.1.1 Detecting Corner Features

We use Shi and Tomasi [7] corner detector that is modified from the Harris corner detector. The Shi corner detector assume the motion can be modeled by Affine, and the motion is small. The assumption is reasonable for this research because we have enough frame rates (15 frames/sec) so that the movement of vehicles is bounded by small range. Given the image patch over the area (u, v), Harris corner detector find corner with shifting it by (x, y). The SSD between these two patches, S is given by

$$S = \sum_u \sum_v (I(u,v) - I(u-x, v-y))^2$$

When I is the intensity of image a position (eg. u,v).
The Harris matrix is found by taking the second derivative of S around (x,y) = (0,0). A is given by

$$A = \begin{vmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{vmatrix}$$

The strength of the corner is determined by 'how much' second derivative there is. The two eigenvalues (λ_1 and λ_2) of A can be used to detect the corner. If the two eigenvalues are large, distinct positive value, the a corner is found.

Shi and Tomasi [7] corner detector based on the Harris corner detector. Instead of simple translation, they use

Affine transformation. Given the image patch over the area (u, v), Shi and Tomasi corner detector find corner with applying Affine transformation(A) and shifting it by (x,y).

$$S = \sum_u \sum_v (I(u, v) - I(A(u, v) - (x, y)))^2$$

We have a 6 by 6 matrix because Affine and transformation have six variables. We find the corner, if the 6 by 6 matrix have large eigenvalues.

2.1.2 Extracting Optical Flow

Lucas and Kanades optical flow [5] is used to match the feature points. Given point (u, v) in image I_t the algorithm finds the point $(u+d_x, v+d_y)$ in image I_{t+1} that minimizes e,

$$e(d_x, d_y) = \sum_{x=u-w}^{u+w} \sum_{y=v-w}^{v+w} (I(x, y) - I(x + d_x, y + d_y))$$

2.1.3 Clustering into the Groups

We cluster a set of flows into a group, if the flows have similar direction and close location. A flow (op) has four features (x, y, v_x, v_y) (x, y are the location of the flow; v_x and v_y are the direction of the flow). Given the two vectors of optical flows $op_1(x', y', v'_x, v'_y)$ and $op_2(x'', y'', v''_x, v''_y)$, Euclidean distance($dist(op_1, op_2)$) is defined by following.

$$\sqrt{(x' - x'')^2 + (y' - y'')^2 + (vx' - vx'')^2 + (vy' - vy'')^2}$$

We design simple nearest neighbor clustering algorithm. Given the a flow (op_i), find the closest flow (op_j). If the $dist(op_i, op_j)$ is less than a threshold value θ_{dist}, group the op_i and op_j into same group. If the case when the op_j is already included in a group G', insert op_i into the group G' ($op_j \in G' \implies op_i \in G'$).

If our camera is mounted in the static ground, there is no additional problem to use the optical flow and this nearest neighbor clustering. All the flows have some biased direction in some case, because our camera is also moving. Whenever there is any gap on the road, the camera has so large movement that the next image has large translation. To prevent this 'bump it out' effect, we find the common biased vector $biased_v(vb_x, vb_y)$ if there is. Thus an optical flow $(op_i(x, y, v_x, v_y))$ has new flow (op'_i) as following.

$$op'_i = (x, y, v_x - vb_x, v_y - vb_y)$$

2.2. Haar-like Feature Detector

We use Haar-like feature detector and AdaBoost to detect the cars in the same direction. Recently Viola and Jones [10, 11] suggest AdaBoost into computer vision. The approach prevails in the face detection because their method is fast and accurate.

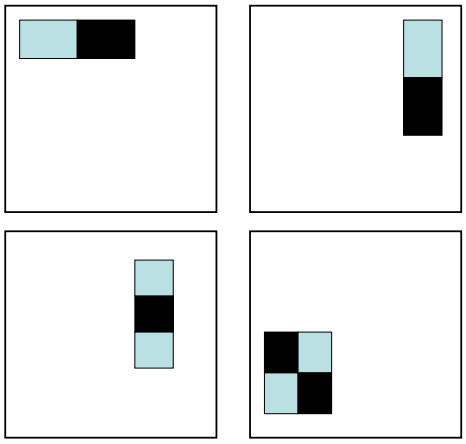

Figure 3: Haar wavelet features are used in this research. A Haar-like feature is defined by one of above rectangle (or variations) and a threshold value. Given an image I, the sum of intensity of dark region and bright region are calculated. If the sum of intensity on the dark region is greater (or less) than the sum of intensity on the bright region by the threshold, the region is determined as positive example. Otherwise, the region is determined as negative example.

2.2.1 Haar-like Feature

We use three kinds of Haar feature as in figure 3. Denote the feature set as $F = f_i | i = 1, ..., N$ and corresponding feature values on images observation z as $V(z) = v_i(z) | i = 1, ..., N$

The number of features even in a 24 by 24 image patch are so huge to calculate them every time. However, we may easily calculate them if we have an accumulated sum of intensity from origin.

$$S_{acc}(i, j) = \sum_{x=0}^{i} \sum_{y=0}^{j} I(i, j)$$

If a rectangle is defined by the region ($[x_{left}, x_{right}] \times [y_{up}, y_{down}]$, the sum of intensity in the rectangle is as following,

$$S_{acc}(x_{right}, y_{down}) - S_{acc}(x_{left}, y_{down}) \\ -S_{acc}(x_{right}, y_{up}) + S_{acc}(x_{left}, y_{up})$$

2.2.2 Learning with AdaBoost

A Haar-like feature is a classifier because a feature can classify the whole group of images into positive image and negative ones. We use the Haar-like feature as a weak classifier of AdaBoost algorithm, as the Viola and Jones did it for face detection. Even if a feature satisfies a simple criteria (the error ratio is less than a guess (0.5)), we can use the

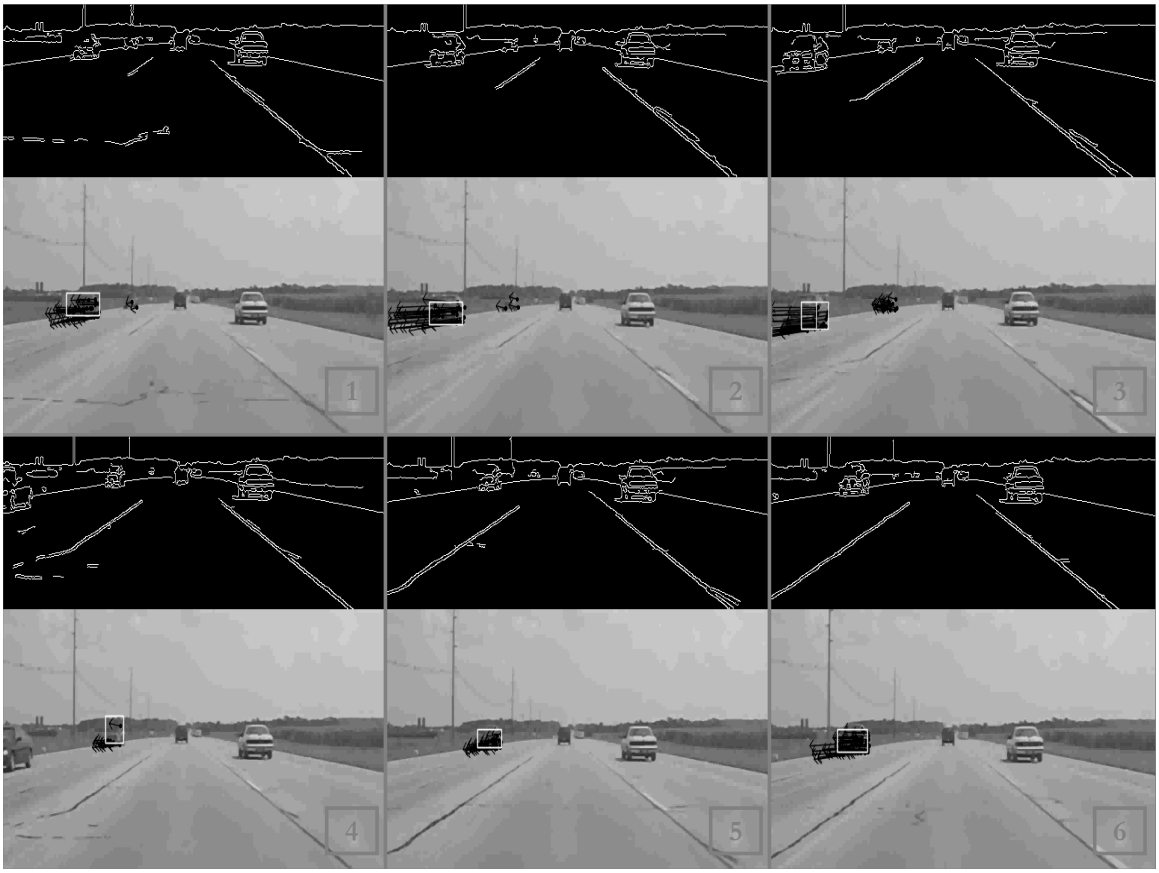

Figure 2: The optical flow is applied to the sequence of images. The black arrows represent the detected optical flows. White rectangles represent the regions that are clustered with nearest neighbor.

feature as a weak classifier. If we have n week classifiers those error ratio are $e_1, e_2, e_3, ..., e_n$ respectively, the total error ratio is bounded by following,

$$\Pi_{t=1}^{T}(2 \cdot \sqrt{e_t(1-e_t)})$$

The total error ratio monotonically decreasing whenever we add weak classifiers, because $2\sqrt{e_i(1-e_i)}$ is less than 1. (To see the detail of proof, please refer [2]. To train the car rear image, we use 334 positive images from two data sets (126 images from 'Cars 1999 (Rear) 2 dataset, CALTECH' and 194 images from 'CBCL CAR DATBASE, MIT' and other 14 images). The 334 positive images are carefully aligned, because Haar-like feature is sensitive to the alignment. We use 500 negative images that are cropped from the normal road such as road, curb, trees and buildings. Figure 4. The strong classifier (which is composed of weak classifiers) is evaluated against training data set and test data set (516 car images from 'Cars 2001(Rear), CALTECH'). In the training set, 88.63% of cars are correctly detected and 11.38% of cars are missed. In the test set, 90.50% of cars are correctly detected and 9.50% of cars are missed. The roc curves are shown in Figure 5.

2.3. Context of the Scene

We apply global context of the scene, because any local detector (the optical flow or Haar-like feature detector) does not count on the global scene. For example, the image detected with Haar-like feature detector have many false positives in Figure 6. Prior information such as the viewpoint (vanishing point and height from the road) and the size of car (eg. 2m) is used to impose context. The information can be used to verify the hypothesis generated from the detectors. To simplify the process, we only consider the size of rectangles (or window) based on the location in the image. Intuitively the window should be large if the window is located on the bottom of image. The window should be

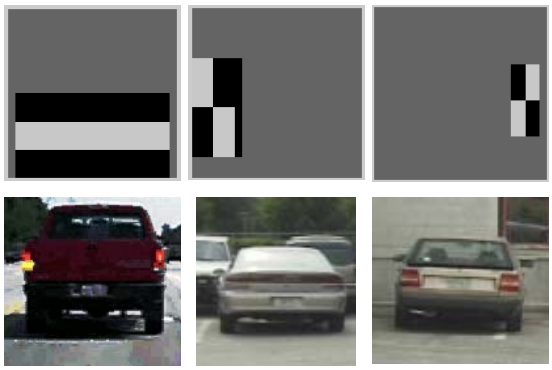

Figure 4: Three of detected Haar-like features are shown at the upper side of figure. Three of sample images are show at the bottom of figure. The leftmost detected feature detects the shadow of car. Other two features show the right edge and left edge of cars.

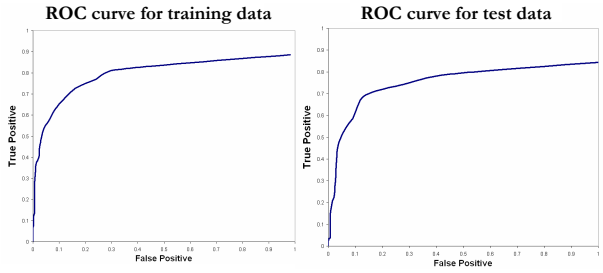

Figure 5: Left graph shows the roc curve for the test data set. Right graph shows the roc curve for the training data set.

small if the window is near the vanishing point. Moreover, there is no window for the car above the vanishing point. The vanishing point (vp_x, vp_y) is found by Hough transform. If the bottom of a rectangle is on the $y = y_{height}$, the size of rectangle is proportional to the $y_{height} - vp_y$ (when $y_{height} > vp_y$). We estimate the probability of size of window based on the location as following,

$$P(Window_{size}|x, y) = P(Window_{size}|y - vp_y)$$

If the probability of a hypotheses (rectangle) is so small, we reject the rectangles.

3. Technical Details

To find the location of the vehicles in the 3D space, we also use other methods. We improve our tracking performance with interpolation between the frames. The Kalman

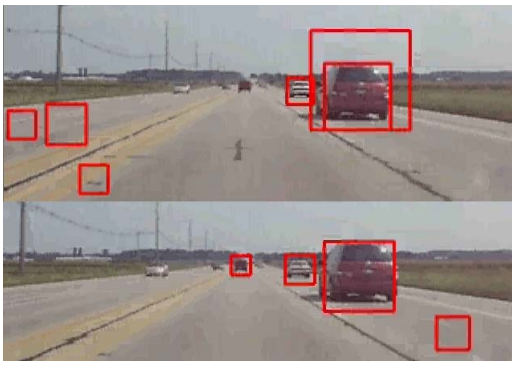

Figure 6: The red rectangles are the regions that are detected by the Haar-like feature detector. The detector finds the location of cars in some case (true positive). However, there are still many false positive rectangles. Some of small rectangles (false positive) can be rejected based on the global context of scene, because we know that the viewpoint and the size of car.

filter is used to smooth the tracking process. The rectangles in the image are mapped onto the 3D geometry.

3.1. Filtering from Multiple Frames

Not only the optical flow but also Haar-like feature detector has many false positives. If we reduce the number of false positive, we also miss many true positives. We cannot restrict number of false positives, because it is very dangerous to have many true negatives. To solve this problem, we maintain the history of detections. We define a flag function as following,

$O(x, y)_t = 1$ x, y is in any detected rectangle at time t
$O(x, y)_t = 0$ otherwise

At time step k, we can lookup the previous n frames. We regard a point has high probability of detection, if a point (x,y) is occupied by any rectangles recently. ($n * f$ of n frames).

$$P(car_{x,y}|k, O(.))_k = 1 \quad if \sum_{i=k-n+1}^{n} O(x, y)_i > n * f$$
$$P(car_{x,y}|k, O(.))_k = 0 \quad otherwise$$

Given the rectangle $[x1, x2] \times [y1, y2]$, we verify the rectangle as a car, if the number of points determined as a car are greater than a certain ratio (eg. 80%). Otherwise, we reject it. The white region in the Figure 8 shows the regions that have $P(car_{x,y}|.)_k = 1$.

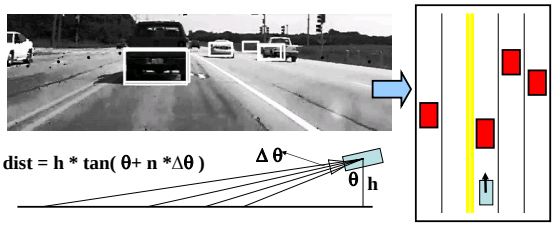

Figure 7: Detected car is mapped onto the 3D space

3.2. The Kalman filter

The detectors do not always work well. That is, a car detected in time t is not be detected in time $t+1$. It is because the Viola and Jones detector is sensitive to the alignment. However we cannot examine all the possible windows, because it takes so long time to prevent realtime image processing. Especially, the detector often misses small cars (long distant one).

To solve this problem, we maintain the list of cars detected at the previous time step. All the variables such as location, size and velocity are maintained with those confidences. If the detectors miss the region, we still generate the region as a car with low confidence. If the detectors find the region, we update the value of variables (location, size and velocity) with high confidence. The Kalman filter is used in this step. One of functions of Kalman filter is smoothing. We have large noise in our image sequences because we are moving while tracking other vehicles. Thus, we have many variances not only in the input image but also in the outputs. However, we can reduce the noise because we maintain the variable based on the previous observation.

3.3. Projection into 3D Space

Finally, the detected and verified rectangles are projected onto the 3D space as the Figure 7.

4. Conclusion

We detect two distinct types of vehicles with 'optical flow' and 'Haar-like feature detector'. We use two algorithms because of the different features between the coming traffics and the vehicles in the same direction. The 'optical flow' is appropriate for the coming traffics because they have relatively salient motion. 'Haar-like feature detector' developed by Viola and Jones is promising to detect the cars in the same direction.

To reduce a number of false positives, we consider the scene context. If we find a vanishing point, we can easily filter many false positives out. Moreover, we look up previous n frames to reduce the false positives out. If the area has

Content	Time/frame(secs)	Time/15 frame(secs)
Read a frame	0.021	0.312
Corner Detection	**0.014**	**0.214**
Hough Transform (Vanishing Point)	**0.029**	**0.429**
Optical Flow	**0.012**	**0.184**
Clustering Flows	**0.001**	**0.008**
Car Verification(Viola Jones) & Kalman Filtering	**0.033**	**0.490**
Write a frame	0.059	0.891
Total Time(inc. image read/write)	0.168	2.527
Total Time (exc. image read/write)	**0.088**	**1.324**

Figure 9: A table that shows the time that takes to process each process. The total time for a frame, it takes 0.168 secs. For the 15 frames, it takes 2.527 secs. However, we may achieve 0.088 secs for a frame, if exclude I/O (the read/write movie file). That takes 1.324 secs for the 15 frames.

no evidence detecting a vehicle, the rectangle (hypotheses) will be rejected.

We maintain the list of cars detected at the previous step. Whenever the detectors fail to find the probable vehicles, we interpolate the candidate in the output. Kalman filter maintain the value and confidence of variables.

We train the Haar-like feature with AdaBoost against 334 images brought from two data sets (1999 CALTECH, CBCL MIT). The performance is evaluated against another 516 images (2001 CALTECH). We achieve the 88.63% of accuracy against test set and 90.50% of accuracy against the training set. Figure 5 shows the roc curve of two results.

The system is fast enough to process 11 frames per seconds. Although Viola's detector takes rather longer time, it is scalable when the training data set increases.[1] In contrast, template-based methods are not scalable to the number of models.

Therefore, the contribution of this work is two things: (1) building a realtime car detection system based on the Haar-like feature detector (2) reducing a number of false positives based on the temporal and spatial context.

Acknowledgement

This paper is a modified version of the final report of CS543 (Computer Vision) taught by Prof. Li Fei-Fei) , Fall 2006.

References

[1] M. Betke, E. Haritaoglu, and L. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *MVA*, 12(2):69–83, 2000. 2

[2] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Compu-*

[1] Source codes can be download at 'reason.cs.uiuc.edu/jaesik/projects/Vehicle_Detection/Jaesik_Choi_src.zip'

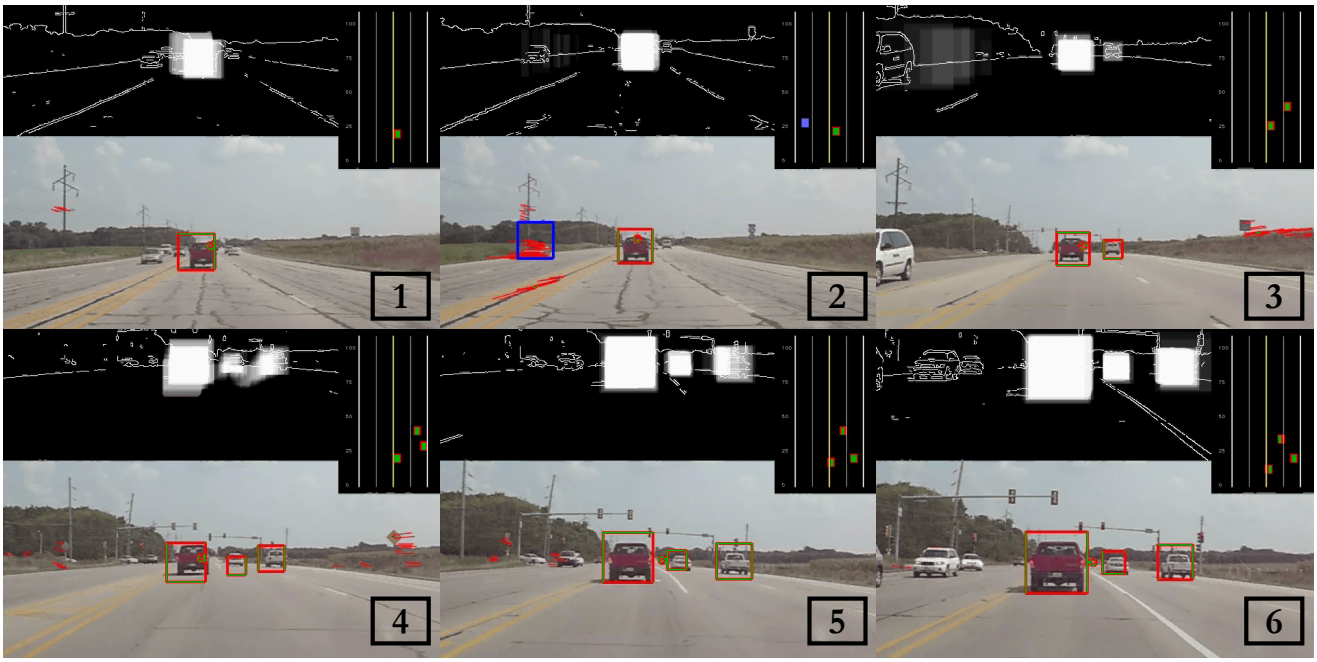

Figure 8: Some of frames, which are extracted from the result movie file. Red rectangles show the cars which are detected by the Viola detector. The blue rectangle is the cars that are detected by the optical flow. White area represents the history of detection. That is the area of high probability of cars. Upper right part shows the locations of cars that is mapped onto 3D.

tational Learning Theory, pages 23–37, London, UK, 1995. Springer-Verlag. 4

[3] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998. 2

[4] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, 2006. 1, 2

[5] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th IJCAI*, pages 674–679, Vancouver, Canada, 1981. 1, 2, 3

[6] R. Miller. On-road vehicle detection: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(5):694–711, 2006. Member-Zehang Sun and Member-George Bebis. 2

[7] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994. 1, 2

[8] M. Stojmenovic. Real time machine learning based car detection in images with fast training. *Mach. Vision Appl.*, 17(3):163–172, 2006. 2

[9] Z. Sun, R. Miller, G. Bebis, and D. DiMeo. A real-time precrash vehicle detection system. In *Proceedings of the 2002 IEEE Workshop on Applications of Computer Vision*, Orlando, FL, Dec. 2002. 2

[10] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision - to appear*, 2002. 3

[11] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance, 2003. 1, 2, 3

[12] J. Wang, X. Chen, and W. Gao. Online selecting discriminative tracking features using particle filter. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 1037–1042, Washington, DC, USA, 2005. IEEE Computer Society. 2

[13] A. Wedel, U. Franke, J. Klappstein, T. Brox, and D. Cremers. Realtime depth estimation and obstacle detection from monocular video. In K. F. et al., editor, *Pattern Recognition (Proc. DAGM)*, volume 4174 of *LNCS*, pages 475–484, Berlin, Germany, September 2006. Springer. 2