Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Vehicle Detection and Tracking in Relatively Crowded Conditions

Wenhao Lu, Shengjin Wang, Xioaqing Ding
Dept. of Electronic Engineering, Tsinghua University
State Key Laboratory of Intelligent Technology and System,
Tsinghua National Laboratory for Information Science and Technology
Beijing , P.R.China
{ luwenhao, wsj, dxq}@ocrserv.ee.tsinghua.edu.cn

Abstract*—Aiming at vehicle detection and tracking problems in video monitoring and controlling system, this paper mainly studies vehicle detection and tracking problems in conditions of high traffic density in daytime. This paper is distinguished by two key contributions. First, we develop an improvement －SEAP(Simple but Efficient After Process) which checks the detection results in an accurate way and is an after process of Adaboost [1] detector which used to detect car in every frame. Second, we propose a tracking algorithm named 4-states tracking algorithm based on Kalman[5] linear filter. Tracking results turn unsteady as traffic density grows higher because of much more false positives and false negatives appear. However, 4-states tracking algorithm can solve this problem in an easy way by introducing FSM (Finite State Machine) into tracking algorithm. Finally, we implement a real-time vehicle detection and tracking system with the upper methods. Experiments give good results in relative crowded Conditions.

***Keywords*—ITS, vehicle detection, adaboost, seap, 4-states tracking, finite state machine**

I. *INTRODUCTION*

Being the most important and fundamental aspects of ITS (Intelligent Transportation System), vehicle detection and tracking becomes more and more important as the number of cars grows in an incredible speed.

Various vehicle detection algorithms have been proposed in the literature. Some algorithms are based on knowledge. They employ prior knowledge, like information about symmetry, color, shadow, textures, to the process of detection. Symmetry is one of the most important signatures of man-made objects. As a result, [13] introduce symmetry into detection and solve problems using Neural Networks. Some algorithms use color information for vehicle segmentation and some others solve the detection problem in color spaces like RGB and L*a*b spaces [14] [15]. Shadow and some horizontal Edge features are useful information for vehicle detection too. As a matter of fact, the front view of vehicle contains many horizontal structures and the shadow of vehicle has a low intensity compared with vehicle's body and surface of the road at most of the time. Some other algorithms use disparity map or IPM (Inverse Perspective Mapping) before trying the upper methods in order to introduce some stereo information into vehicle detection. There's many algorithms are based on vehicle movements. But, all the algorithms mentioned above are not robust to clutter scenes because they don't get the characteristic feature of vehicles. Especially when interactions of vehicles appear in the scene, algorithms using symmetry, color and other features turn less discriminating because it is hard to separate 2 or 3 interacted cars with these simple features.

The most popular vehicle detection algorithm is based on statistical learning [6] [7] [8]. In statistical learning algorithms, vehicle detection is treated as a two-class pattern classification problem. These algorithms try to learn vehicles' features by training computer with large amount of vehicle images and non-vehicle images. Adaboost detector and haar-like features succeeded in face detection [1]. Therefore, it is reasonable to introduce Adaboost and haar-like features to vehicle detection. In our experiment, it is more robust then algorithms which use simple features, but detection results are not stable in the image series. For example, car A is detected in 1-6 and 10-20 frames but is missed in 7-9 frames(false negatives in some frames), or car A's size is detected as 30*30 in the first frame but changed to 60*60 in the second frame. These occasions appear because Adaboost performances well only when the variances in class are small and the variances between classes are quite large [3]. So it is necessary to add an after process module to optimize the detection results. It is why we designed SEAP method which will be explained in detail in Section 3. On the other hand, we need some other algorithms to help us link the detection results of every frame, to solve the false negatives problem talked above. As a result, we should employ tracking algorithm as well. There has been many effective algorithm provided on object tracking, including Kalman filter [9], Extended Kalman filter [10], Particle filter [11], Meanshift [12], etc. Among all tracking methods, Kalman Linear Filter [5] is the most widely used tracking algorithm in the problems like tracking vehicles because the size and position of every car change linearly [2]. But as the traffic density grows higher, much more false positives and false negatives that are mentioned above appear in the detection results, and more interactions between cars should also be considered. Tracking algorithm simply using Kalman filter gives unstable results that is unbearable for normal usage when the road is crowded.

In order to solve the problems mentioned above, we implement a complete real-time car detecting and tracking system using an improved detector and a tracker called 4-states tracker. There are two key contributions in our system. The

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

first contribution is an improvement — SEAP(Simple but Efficient After Process) which is an after process of Adaboost [1] detector and checks the detecting results in an accurate way. We select an Adaboost detector that cascades less stages which performances high false positives but low false negatives in car detection. And then, we add our after process——SEAP following the Adaboost detector. SEAP moves away many false positives using Harris corner detecting first, then finds the cars' accurate position by detecting the shadow area in the front of every car. This after process can be computed rapidly since it just computes in the areas that surround every detected car. The second contribution is a new approach called 4-states tracking algorithm. Our goal is to find a way that can give stable tracking results when more false positives and false negatives appear in the scene. Therefore, FSM (Finite State Machine) is introduced in our tracking algorithm. We design four kinds of states called 'Detecting state', 'Preparing Locking state', 'Locking state' and 'Preparing Unlocking state' for a single car's tracking process.

The rest of this paper is organized as follows: Section 2 gives a specific explanation of SEAP method. Section 3 discusses the 4-states tracking algorithm in detail. In section 4 we show the experimental results and analysis.

II. SYSTEM STRUCTURE

The system described in this paper is introduced for traffic flow surveillance in intersection. As shown in Figure 1, the system is composed by three parts: moving objects detecting and tracking, detection regions setting, traffic movements detecting. In the first part, the moving objects are accurately In our system shown in Figure 1, Adaboost detector searched objects in each frame. Then the output information is improved by SEAP in the next module. After the detection and improvement, the system sends all of the information into 4-States tracker. There are many FSMs in the tracker and each FSM will tracks one detected object using 4-States algorithm. If the tracker finds a new object, new FSM will be assigned.

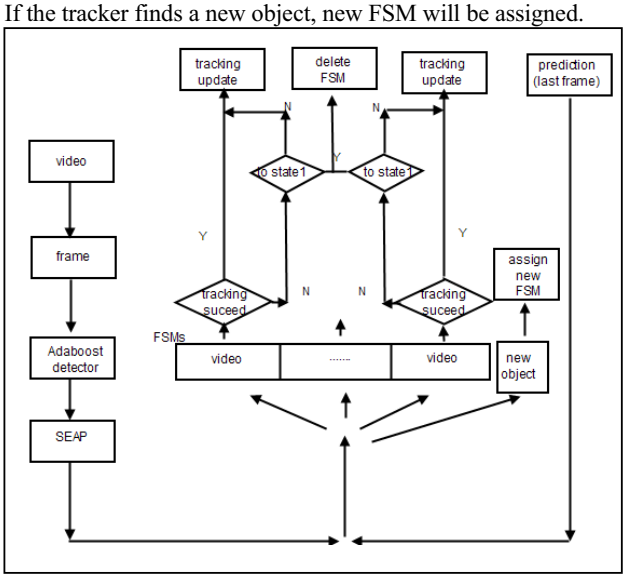


Figure 1. *System Flow Chart.*

III. SEAP METHOD

Adaboost detector has a cascade structure. The false positives probability falls down and false negatives probability rises up as the number of stages grows. It is a trade-off between false negatives and false positives. When facing the problem of car detection in crowded roads, Adaboost detector can't find the perfect trade-off, the results have either too many false positives or false negatives, neither meets our demands. In order to overcome the problem, we add an after process module(SEAP) behind a normal Adaboost detector. In this case the Adaboost detector has less stages and remains low false negatives probability.

SEAP moves away some false positives first. When we analyze the feature selected by Adaboost detector, we find the most important Haar-like feature of a car is the vertical gradient in the shadow area at each car's front (see Figure 2.(a). Another important Haar-like feature is the vertical gradient shown in Figure 2.b.as a result, false positives rise when the Adaboost detector finds apparent vertical gradient in a detecting window. We can find that lots of false positives take the car window for car or just take a crowd of cars for one object (see Figure 3).

Figure 2. *The specific Haar-like feature.*

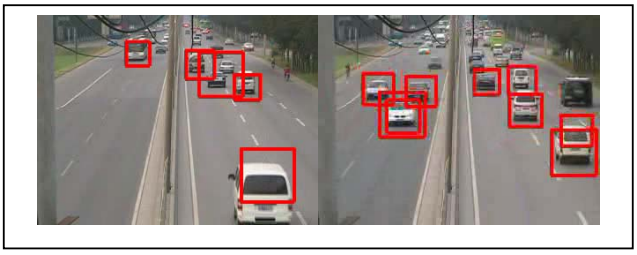


Figure 3. *Lots of false positives take the car window or just take a crowd of cars.*

Based on this observation, we designed the false positive checking method below:

- *Check all the windows ($w_i \sim w_j$) that detected by Adaboost detector, if w_i interacts with w_j and* $area(w_i) \cap area(w_j) \geq \frac{1}{2} area(w_i)$ *(* $area(w_i) \cap area(w_j) \geq \frac{1}{2} area(w_j)$ *), start checking false positive in these two windows.*

- *Detecting corner using Harris corner detector [4] in w_i, w_j.*
- *Calculate the center of gravity based on these corner points (or we can some other statistics to present a car), compared two center to decide which one is a false positive. Normally, we choose the one that its center of gravity is closer to the center of the window.*

$$a = \frac{1}{y_{\max} - y_{\min}} \sum_{shadow\ area} (x_{\max} - x_{\min})$$

$$x_0 = \frac{1}{y_{\max} - y_{\min}} \sum_{shadow\ area} x_{\min}$$

$$y_0 = y_{\max}$$

$$window(x, y) =$$

$$\begin{cases} 1 & \left| x - x_0 - a/2 \right| \le a * \rho \ and \ \left| y - y_0 + a/2 \right| \le a * \rho \\ 0 & else \end{cases}$$

where ρ is a parameter for adjusting the final window size.

After moving away the false positives, SEAP needs to find the accurate position of each car. As we have mentioned above, the most important Haar-like feature of a car is the vertical gradient in the shadow area at each car's front (see Figure 2.(a)). We suppose that a car's contour looks much like a square in the front or from its back. So we search the obvious vertical gradient in each detected window and measure its average width, then take it as the square edge length (see Figure 4). This method makes the size of a car stable, so that we can use the size information in tracking algorithm.

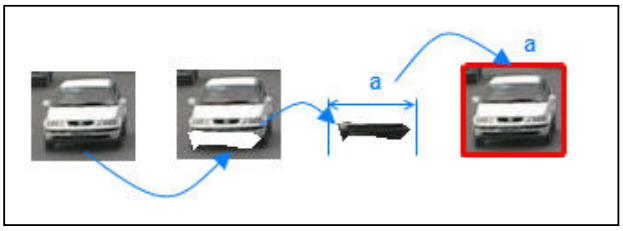


Figure 4. *Accurate object size using shadow information.*

SEAP is the module between Adaboost detector and 4-states tracker. Its function is to make the output of Adaboost detector more stable and accurate. SEAP finishes its job in an easy way, and it is efficient in ordinary occasions though it seems not smart enough yet. In our experiments, it cut off over 85% false positives probability with no apparent change in detection rates.

IV. 4-STATES TRACKING ALGORITHM ECTION

When traffic density grows higher, more false positives and false negatives appear in detector's results, and the interactions between cars can't be ignored. All these problems lead to difficulties for tracking cars. Traditional tracking algorithm divided the whole frame into two parts: detecting area and tracking area. A traditional tracker only track a detected object in tracking area. It has a strong assumption which is the cars will be detected in the detecting area. Actually, lots of uncertainties makes it's possible to detect an object for the first time everywhere in a frame (see Table1. maybe due to some false negatives and false positives, maybe when a small car finally speed up and appears in the camera from behind a large bus).

TABLE I. SEVERAL OCCASIONS OF A CAR'S APPEAR AND DISAPPEAR

	OCCASIONS	**DESCRIBE IN DETAIL**
Car disappear	Move out of camera	Disappear at image edges
	Car interaction	One car moves behind the other car
	False negatives	False negative won't last for many frames
	Disappear of false positives	
Car appear	Move into camera	Appear at image edges
	Car interaction	One car appears from behind the other car
	False positives	False positives won't last for many frames

Think of how people get to know a new friend. When person D (Detector) meet person O (Object) for the first time, D don't know anything about O. Is O a student or a waiter? D has no idea. As they meet each other frequently in the following days, D knows O's living customs and O's uncertainty becomes smaller and smaller. Finally D confirms that O is a student based on his continually observation. In fact, D still can forget O slowly if O never shows up in D's life, but it is not an easy process. The process seems likely in our car tracking problem. We try to use the relations in a series of frames to help us confirm whether an object is a car and make sure its track won't be missed easily.

So we propose FSM into our car tracking algorithm and define four kinds of states:

State 1: Detecting state. We define the input and output of each state first. When we detect a new car or detect an old car in its small predicted area, the input of this state is 1, otherwise, the input value is 0. When we confirm the object in a car's predicted area is the car, the output of this state is 1, otherwise is 0. At detecting state, the FSM turns to Preparing Locking state when the input is 1 (detect a new car), or remains at Detecting state if the input is 0.

State 2: Preparing Locking state. This state is an N times' confirmation. Counter1 starts when the FSM turns into this state which counts the number of frames in which the FSM stays in this state. Counter2 starts when the state's begins to output 0 and Counter2 will be back to 0 when the output turns to 1 before Counter 2 counts to M. Counter1 will stop when Counter2 counts to M or counts to N itself (N>=M). Actually, there are $M * N - \frac{1}{2} * (M-1) * (M-2)$ sub-states, they are present as follow.

Sub-state1: counter1 = 1, counter2 = 0;

Sub-state2: counter1 = 1, counter2 = 1;

Sub-state3: counter1 = 2, counter2 = 0;

Sub-state4: counter1 = 2, counter2 = 1;

Sub-state5: counter1 = 2, counter2 = 2;

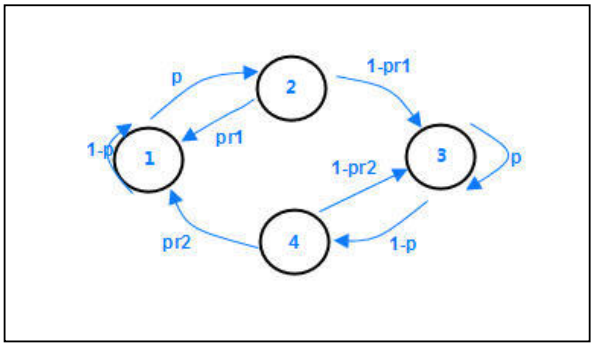
Sub-state6: counter1 = 3, counter2 = 0;

…….

If Counter1 gets to N, we turn all the output in the former N frames to 1, otherwise to 0.

State 3: Locking state. In this state, we are quite sure that the detected object is a not a false positive. The state output is 1 no matter the input is 1 or 0. The FSM remains in Locking state when the input is 1 but turns into Preparing Unlocking state if the input is 0.

State 4: Preparing Unlocking state. In this state, the FSM still think the detected object is not a false positive, but it starts to suspect. It is a Q times' confirmation which likes the Preparing Locking. The difference is we are confirming whether the object is a false positive instead of whether it is an expected car in this state. Counter3 starts when the FSM turns into this state and will be back to 0 if it gets to Q or re-tracks the car before Counter3 turns to Q. It is also a set of sub-states like the Preparing Locking state and has totally Q states.



We can analyze the algorithm using Markov chain theory (see Figure 5).

Figure 5. *Transport graph.*

We define the probability of states 1 to 4 as $\tilde{\mu} = (\mu_1, \mu_2, \mu_3, \mu_4)$, while the transport matrix is

$$T = \begin{bmatrix} 1-p & p & 0 & 0 \\ p_{r1} & 0 & 1-p_{r1} & 0 \\ 0 & 0 & p & 1-p \\ p_{r2} & 0 & 1-p_{r2} & 0 \end{bmatrix} \quad (1)$$

p is the detect rate of the detection.

We assume that $N/2 \le M < N$, then obtain

$$\mathrm{P}_{r1} = \sum_{i=N-M}^{N-M-1} C_{N-M+1}^{i-M+1} p^{N-i} (1-p)^i \quad (2)$$

$$\mathrm{P}_{r2} = (1-p)^Q \quad (3)$$

We can get the probability of

$\tilde{\mu} = (\mu_1, \mu_2, \mu_3, \mu_4)$

by the following steps.

$$\begin{cases} \tilde{\mu} = \tilde{\mu} * T \\ \mu_1 + \mu_2 + \mu_3 + \mu_4 = 1 \end{cases} \Rightarrow$$

$$\begin{cases} \mu_1 : \mu_2 : \mu_3 : \mu_4 = \\ p_{r2}(1-p) : p * p_{r2}(1-p) : p(1-p_{r1}) : p(1-p_{r1})(1-p) \\ \mu_1 = \dfrac{p_{r2}(1-p)}{p_{r2}(1-p)(1+p) + p(1-p_{r1})(2-p)} \end{cases} \quad (4)$$

The probability of μ_1 tells us how frequency an object's FSM turns to state 1. A low μ_1 stands for less tracking-loss conditions. And Figure 6 shows the $\mu_1 - p$ curve. As we can see, μ_1 turns rather low while detect rate is not very high.

As we can see, state 2 takes quite a long time to confirm about the object, it keeps the response of false positives in a low level. State 3 and state 4 make our system be possible to handle the common false negatives in frames. The hops between these 4 states are shown in Figure 8

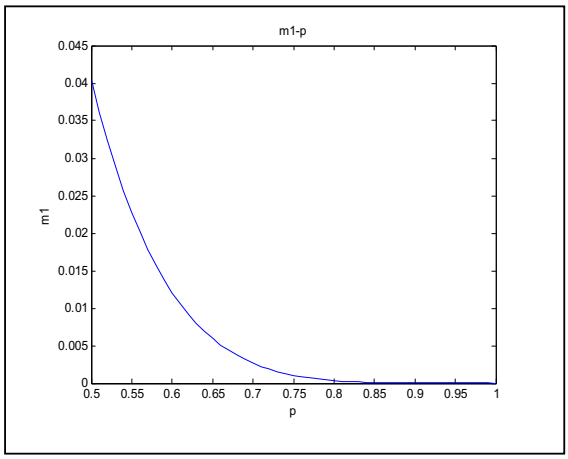


Figure 6. μ_1 - p *Curve(N=12,M=6,Q=4).*

V. EXPERIMENTAL RESULT

In our system, we designed a 10 stages cascaded classifier following with a SEAP module and a 4-state tracker (N=12,

M=6, Q = 6). The images for training are 24*24 pixels, and there are totally 1600 positive samples and 2000 negative

TABLE II. EXPERIMENT FOR TESTING SEAP MODULE

	Frame number	Average car number	Without SEAP		SEAP added	
			Detect probability	*False positives number*	*Detect probability*	*False positives number*
1	230-240	3	90.0%	7	86.7%	2
	280-290	4	76.3%	16	84.7%	1
2	5-15	11	86.1%	6	88.0%	1
	80-90	7	98.6%	30	97.1%	0
3	230-240	7	83.4%	10	85.5%	1
	370-380	8	70.0%	15	68.8%	0

Frame number	With traditional tracker			With 4-states tracker		
	Detect probability	*False positives number*	*Track failure number*	*Detect probability*	*False positives number*	*Track failure number*
100-110	93.3%	5	3	93.3%	0	0
150-160	96.0%	0	2	100%	0	0
410-420	94.3%	2	3	92.9%	0	0

TABLE III. EXPERIMENT FOR TESTING 4-STATE TRACKER WITH VIDEO 3

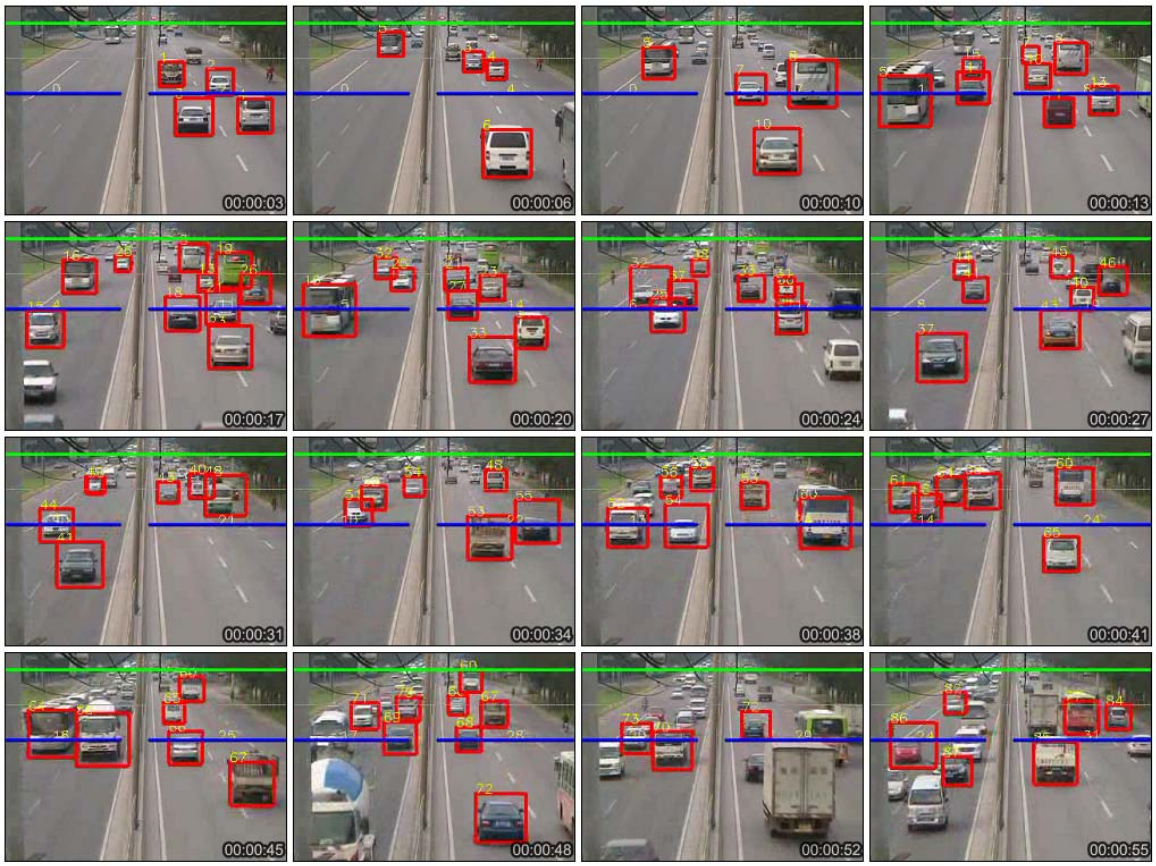


Figure 7. *Some detection and tracking results.*

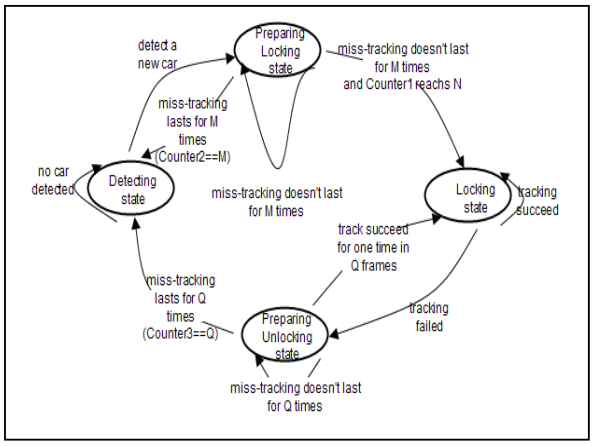


Figure 8. *Hops between the 4 states.*

samples. The testing videos are 320*240 pixels and the frame-rate is 25 fps. Testing results are as follow Tables:

Statistic shows, the average time consumed on one frame is 45.8ms, the maximum and minimum times consumed on one frame are 78ms and 15ms.So our system almost is a real-time system.

Analyze: as the data shows, SEAP can keep the output of Adaboost detector in a fairly low false positive probability so that helps our 4-states tracker to solve tracking problem more easily. And 4-state tracker makes the false positive probability lower with almost no cost. As we have state, our system makes the detecting and tracking results stable and usable. In most of the conditions, our system lowers the false positives probability while keeps the detect probability or makes it grow a little bit higher. Only in a few conditions, the detect probability falls down obviously in our system.

VI. CONCLUSION

In this paper, we implement a complete real-time car detecting and tracking system with the upper methods and experiments with our system give a more stable and accurate result compared with the old algorithm. We discussed detecting and tracking cars in a relatively crowded road which has lots of false positives and false negatives in Adaboost detector's output. We designed SEAP method and 4-state tracking algorithm which makes our detecting and tracking results more accurate and stable. As experiments shows, our methods are effective and efficient at the same time.

ACKNOWLEDGMENT

This work is supported by The National High Technology Research and Development Program of China (863 program) under contract No. 2006AA01Z115 and the National Basic Research Program of China (973 program) under Grant No. 2007CB311004.

REFERENCES

[1] *P. Viola and M. Jone, "Rapid object detection using a boosted cascade of simple features," in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2001.*

[2] *zhi Zeng, "Real-time car detecting and tracking based on road area analysis" Dept.EE Tsinghua University's master degree thesis 2006*

[3] *Barlett P, Baxter K and Mason L. Improved generalization through explicit optimization of margins. Machine Learning, 38:243-255, 2000*

[4] *yu jin Zhang, "Image Analysis" ,pp. 139-150,2007*

[5] *you wei Zhang, "Weiner filter and Kalman filter introduction", 1980*

[6] *C. Papageorgiou, M. Oren and T. Poggio, "A general framework for object detection," in Proceeding of IEEE International Conference on Computer Vision. pp. 555-562, 1998.*

[7] *S.Wender and O.Loehlein, "A cascade detector approach applied to vehicle occupant Monitoring with an omnidirectional camera," in Proceeding of IEEE International Conference on Intelligent Vehicles, pp. 345-350, 2004.*

[8] *S. Agarwal, A. Awan and D. Roth, "Learning to Detect Objects in Images via a Sparse, Part-Based Representation," IEEE Transaction on Pattern Recognition and Machine Intelligence. 26(11), pp. 1475-1490, 2004*

[9] *G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Course Notes, ACM SIGGRAPH, 2001*

[10] *S. J. Julier and J .K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in Proceeding of International Symposium on Aerospace/Defense Sensing, Simul. and Controls. 1997*

[11] *M. Isard and A. Blake, "CONDENSATION - conditional density propagation for visual tracking", International Journal on Computer Vision, 29(1), pp. 5-28, 1998*

[12] *D. Comaniciu V. Ramesh and P. Meer, "Real-Time Tracking of Non-Rigid Objects Using Mean Shift," in Proceeding of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 142-149, 2000.*

[13] *W. von Seelen, C. Curio, J. Gayko, U. Handmann, and T. Kalinke, "Scene Analysis and Organization of Behavior in Driver Assistance Systems," Proc. IEEE Int'l Conf. Image Processing, pp. 524-527, 2000.*

[14] *S.D. Buluswar and B.A. Draper, "Color Machine Vision for Autonomous Vehicles," Int'l J. Eng. Applications of Artificial Intelligence, vol. 1, no. 2, pp. 245-256, 1998.*

[15] *M. Xie, and C. Laugier, "Color Modeling by Spherical Influence Field in Sensing Driving Environment," Proc. IEEE Intelligent VehicleSymp.,pp.249-254,2000*