# Real Time Car Detection and Tracking in Mobile Devices

Ayan Acharya
*Department of ECE*
*University of Texas at Austin, USA*
*aacharya@utexas.edu*

Jangwon Lee, An Chen
*Office of the Chief Scientist*
*Qualcomm Inc., San Diego, USA*
*jangwonl@qualcomm.com, anc@qualcomm.com*

***Abstract*—The paper presents a real time car detection and tracking system that can be used in a mobile device equipped with a camera (*e.g.* handset, tablet etc.). The underlying detector is implemented using a variant of the AdaBoost [5], and the tracker is built on Lucas-Kanade optical flow algorithm. The uniqueness of the work lies in the simplicity and the portability of the framework that can assist drivers in real time without much computation effort.**

***Keywords*-detection; tracking; boosting; android;**

## I. Introduction

Driver safety and automatic navigation has been an active area of research since the past decade. Recent developments in hardware have enabled numerous computer vision algorithms to be implemented in real time. Efforts have been made for car detection and tracking using simple segmentation and tracking algorithms or some simple probabilistic classifiers [2, 3]. Unfortunately, all these implementations are based on PC and, to our knowledge, there does not exist any work on limited processing capabilities such as a mobile device.

In this paper, we propose a real time car detection and tracking system based on monocular vision that can run in real time on any mobile device equipped with camera. We also implement the proposed system on Android smartphone/tablet. So, a user can simply install the application on the device which is mounted on the dash-board of the car.

## II. Detection and Tracking Framework

The system block diagram of the proposed framework is shown in Figure 1. The detector is a classifier which scans through the entire image. In mobile device implementation, the detector is applied every few frames or when the "quality" of the tracker goes below certain threshold. Below, the detector and the tracker are discussed in detail.

### *A. Detection*

The detector is a cascaded classifier built from an ensemble of decision stumps following the works of [1, 7]. Essentially, each stage of the cascade is an AdaBoost [5] classifier where each weak learner corresponds to Local Binary Pattern features (**LBP** – [1]). The cascaded architecture speeds up the detection process where negative examples are discarded in the initial few stages and the computation effort is concentrated more on patches hard to classify. In [1, 7], this kind of classifier was used to detect human faces. In this work, we extend such classifier to detect back of the cars.

As suggested in [1], **LBP** features are useful for detecting objects which have certain groove like parts in their appearance. We built our own training database partly by crawling from web and partly by capturing video. This yielded a training database of 2000 positive examples all of which constitute back of the cars viewed with an angle close to 0 degree with some background. In our collected images, we had an average aspect ratio of $24 \times 20$ and all the training images are scaled to the exact same size (in pixels) while building the cascaded classifier. In test phase, the detector cannot detect any car which occupies less than the dimension $24 \times 20$ in the image [7]. The negative examples for training the classifiers (approximately 2500) are carefully chosen from the MIT LabelMe database which consist of images of skies, trees, empty roads and buildings.

The number of stages in the cascaded architecture plays a crucial role in making a trade-off between accuracy and prediction speed. If more number of stages are used, more effort is spent in classifying positive examples in the cascaded architecture and the prediction quality becomes superior. However, for real time on-board vehicle implementation, prediction speed is of utmost importance. Therefore, to make a trade-off between accuracy and speed, we trained the cascaded architecture with 10 stages with a maximum false alarm rate of $0.2$ and a minimum hit rate of $0.995$. To adjust for slight variations in the viewing angle, we also rotate the training images randomly around horizontal and vertical axis within a window of $\pm 2.5°$. In a typical face detection framework, anywhere between 30 to 40 stages are used to achieve an accuracy of around 98% [7]. However, with the current training setting, we are able to achieve an accuracy as high as 94% during test phase. In detection phase, the classifier scans through the entire image and tries to find a match for patches similar to positive training examples.

### *B. Tracking*

To avoid running classifier every single frame, the Lucas-Kanade optical flow based tracker [8] is employed. The tracker is initiated with the nearest identified car from the

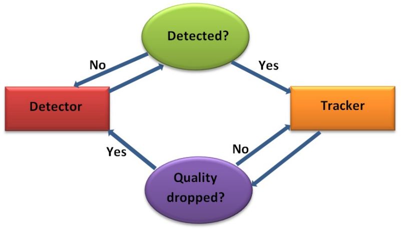

Figure 1. Overview of the Proposed Framework.

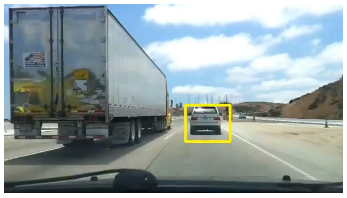

Figure 2. Sample Detection 1.

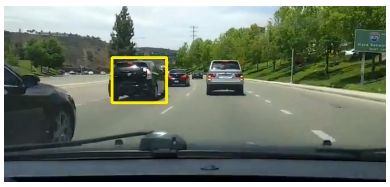

Figure 3. Sample Detection 2.

detector and is used to track the car till the "quality" of tracking goes down a certain threshold. The quality of the tracking process is controlled by two factors: i) the number of corners in the current region of interest (**ROI**) in the current frame, and ii) the relative symmetry of the current **ROI** w.r.t the **ROI** handed over by the detector. The symmetry of any ROI with height $h$ and width $w$ is defined a follows:

$$S = \sum_{y=1}^{h} \frac{\int_0^w H_e^2(x,y)dx - \int_0^w H_o^2(x,y)dx}{\int_0^w H_e^2(x,y)dx + \int_0^w H_o^2(x,y)dx}. \qquad (1)$$

Here, $H_e(x,y) = (I(x,y) + I(-x,y))/2$ and $H_o(x,y) = (I(x,y) - I(-x,y))/2$, where $I(x,y)$ being the intensity of the $(x,y)^{\text{th}}$ pixel. When the tracking process is initiated, the detector hands over an **ROI** to the tracker. We keep record of the symmetry of this **ROI** and compare the symmetry of the tracked **ROI**s in the next frames. If either this ratio goes below certain threshold or the number of detected corners go below some threshold, the tracker ends and the detector is triggered.

## III. Experimental Results

We implemented the proposed framework on both PC and Android device. We show some sample detection and tracking results in Fig. 2 and 3 where the tracked car is marked by a yellow box.

The classifier (trained with 2000 positive examples and 2500 negative examples and consisting of 10 stages) is employed to identify five different annotated videos (neither of which are used in the training process). This yielded an average precision of 94.74% and an average recall of 88.24%.

In PC implementation, the system can achieve a frame rate of up to approximately 120. In Android implementation, the average frame rate is approximately 8, which is a bit low. However, this is primarily due to the inherent slow access of Java API of the camera from Android devices. With empty loop (i.e. with no detection and tracking implemented), the maximum achievable frame rate was 10. This indicates that around 2 frames per second worth computation effort are being used for our framework. We also tried to implement the framework (AdaBoost classifier with between 30 to 40 stages) on the Android device and found out that the maximum achievable frame rate is 4, which cannot be used in real time environments.

## IV. Conclusion and Future Work

For car detection and tracking in real time environments, we proposed a framework which achieve a good real time performance while requiring less computation. There are many possible ways to improve upon our current model. Firstly, we would like to develop a system that can detect different types of vehicles as well as pedestrians and signals without employing a separate classifier for each of them and requiring less training data – using a technique called multi-task learning [4]. Secondly, the prediction time can further be reduced by incorporating the complexity of the weak learners in the training phase [6].

## References

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:20372041, 2006.

[2] A. Petrovskaya and and S. Thrun. Modelbasedvehicledetectionand tracking for autonomous urban driving. *Autonomous Robots, Special Issue: Selected papers from Robotics: Science and Systems 2008*, 26(23):123139, April 2009.

[3] J. Arrospide, L. Salgado, and M. Nieto. Video analysis-based vehicle detection and tracking using an mcmc sampling framework. *EURASIP J. Adv. Sig. Proc.*, 2012:2, 2012.

[4] R. Caruana. Multitask learning. *Machine Learning*, 28:4175, July 1997.

[5] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *In EuroCOLT 95*, pages 2337. Springer-Verlag, 1995.

[6] A. Grubb and D. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. *Journal of Machine Learning Research - Proceedings Track*, 22:458466, 2012.

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

[8] J.Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.