

**VEHICLE TRACKING BY FUSING MONOCULAR CAMERA
AND VEHICLE TO VEHICLE COMMUNICATION ON A REAL TIME BASIS**

**(KAMERA VE ARAÇ-ARAÇ HABERLEŞMESİ BİRLEŞİMLİ
GERÇEK ZAMANLI ARAÇ TAKİBİ)**

by

Mustafa TEKELİ, B.S.

Thesis

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Feb 2017

This is to certify that the thesis entitled

**VEHICLE TRACKING BY FUSING MONOCULAR CAMERA
AND VEHICLE TO VEHICLE COMMUNICATION ON A REAL-TIME BASIS**

prepared by **Mustafa TEKELİ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering** at the **Galatasaray University** is approved by the

Committee:

Prof. Dr. Tankut ACARMAN (Supervisor)

Department of Computer Engineering

Galatasaray University -----

Assist. Prof. Dr. Murat AKIN

Department of Computer Engineering

Galatasaray University -----

Prof. Dr. Mehmet Turan SÖYLEMEZ

Control and Automation Engineering Department

Istanbul Technical University -----

Date: -----

ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis advisor Prof. Dr. Tankut ACARMAN for the continuous guidance and endless effort throughout my research as well as my whole master program. He consistently allowed this thesis to be my own work, but steered me in the right direction whenever he thought I needed it.

I also thank Mr. Muhammed ALYÜRÜK, who is the general manager of my company (İSBAK A.Ş.) for making this work possible by providing support and giving importance to academic researches.

Finally yet importantly, I would like to thank my family that has always been supporting me during my whole education life and Deniz HACIHALİL for proofreading and advices.

February 2017

Mustafa TEKELİ

TABLE OF CONTENTS

LIST OF SYMBOLS	v
LIST OF FIGURES	vii
LIST OF TABLES	viii
1. INTRODUCTION	1
2. LITERATURE REVIEW	3
3. MATERIALS AND METHODS	8
3.1. Vehicle Detection.....	10
3.2. Distance and Relative Speed Estimation	14
3.3. Tracking	17
3.4. Sensor Fusion.....	19
4. EXPERIMENTAL STUDY	23
5. RESULTS & CONCLUSIONS.....	41
REFERENCES.....	43
BIOGRAPHICAL SKETCH.....	46

LIST OF SYMBOLS

V2V	: Vehicle to Vehicle Communication
V2I	: Vehicle to Infrastructure Communication
DSRC	: Dedicated Short-Range Communication
UWB	: Ultra Wide Band
IMU	: Inertial Measurement Unit
LIDAR	: Laser Imaging Detection and Ranging
ROI	: Region of Interest
CPU	: Central Processing Unit
GPU	: Graphics Processing Unit
ms	: Milliseconds
ITS	: Intelligent Transportation Systems
PC	: Personal Computer
Hz	: Hertz (Unit of frequency)
SIFT	: Scale-Invariant Feature Transform
PCA	: Principle Component Analysis
NN	: Neural Networks
SVM	: Support Vector Machine
TP	: True Positive
FP	: False Positive
FN	: False Negative
ER	: Error
TPR	: True Positive Rate (Recall)
PPV	: Positive Predictive Value (Precision)
Kmph	: Kilometer per Hour
UIUC	: University of Illinois at Urbana-Champaign
GPS	: Global Positioning System

ECEF : Earth-centered Earth-fixed
LAN : Local Area Network
IEEE : The Institute of Electrical and Electronics Engineers
ISM : Industrial Scientific Medical Band
CAM : Cooperative Awareness Message
GHz : Giga Hertz
MHz : Mega Hertz



LIST OF FIGURES

Figure 2.1: Vehicle Validation.....	4
Figure 2.2: HV using wavelet features	7
Figure 3.1: Integral image method.....	11
Figure 3.2: Sample images from UIUC image database	13
Figure 3.3: Detected car on test image of UIUC image database.....	13
Figure 3.4: Detection time of cars in our trained system.....	14
Figure 3.5: Image analysis for the lane marker detection.....	15
Figure 3.6: Pixel based inter-vehicle distance estimation in 2D image plane	16
Figure 3.7: The Forward-Backward error.....	18
Figure 3.8: The Median Flow algorithm.....	19
Figure 3.9: Detection and tracking algorithm of our application	20
Figure 3.10: Particle filter algorithm.	22
Figure 4.1: Test system for data collecting.....	24
Figure 4.2: Data collection method.....	25
Figure 4.3: IMU unit orientation and vehicle drive direction.....	26
Figure 4.4: Sample from KITTI vision benchmark.	27
Figure 4.5: Birds-eye-view window.	29
Figure 4.6: Detected vehicles.....	31
Figure 4.7: Detected vehicles and birds-eye-view projection.....	32
Figure 4.8: Sensor fusion view.	33
Figure 4.9: Vision sensor detections.....	34
Figure 4.10: Out-of-sight scenario.....	35
Figure 4.11: Particle filter result.	35
Figure 4.12: Path of the road data used in experiment.	36
Figure 4.13: Hardware features of NVIDIA Jetson TK1 board.	37
Figure 4.14: Canny edge detection and Hough Transform performance on resized (320x240) image (left) vs the original (3888x2592) size (right).....	38

Figure 4.15: Canny edge detection and Hough Transform performance on resized image (320x240)..... 39

Figure 4.16: Canny edge detection and Hough Transform performance on original image (3888x2592)..... 39

Figure 4.17: Output of CPU vs GPU line detection test on the highway image 40



LIST OF TABLES

Table 2.1: Execution time on highways.....	5
Table 2.2: Average processing time of each algorithm	6
Table 3.1: List of parameters in IEEE 802.11p.....	10
Table 3.2: Used Haar features.....	10
Table 4.1: CAM samples received from the data sender in IEEE 802.11p	26
Table 4.2: IMU samples.....	27
Table 4.3: Comparison of TPR (or Recall), PPV (or Precision) of KITTI vision benchmark and our road data from Istanbul	28
Table 4.4: Distance and speed comparison of camera and V2V sensors for a detected vehicle.....	28
Table 4.5: Summary of our collected data.....	36

ABSTRACT

Nowadays autonomous cars have already started to be used in our daily lives. They are the new generation vehicles, which are able to operate themselves under certain conditions without the need of a driver's response. They can follow the lane markers and keep themselves inside the lane, track the vehicles around and decide and perform a break or even change the present lane where necessary. Driver assistance systems are the underlying technology of these vehicles. It enables them to analyze the road and traffic conditions in order to take these kinds of necessary actions. This makes the task of detection of objects at the road such as vehicles, lane markers, guardrails and even pedestrians walking by highly important. For this reason, it is also very substantial to be able to operate in real time conditions.

Current studies focused on this area are mostly based on vision sensor, LIDAR, microwave sensors etc. However, they alone suffer from conditions such as high variety of targets, lighting fluctuations, short range, requirement of high process power, cost etc. In this work, we introduce a monocular camera based system, which is enhanced with the IEEE 802.11p vehicle-to-vehicle communication standard. By the help of IEEE 802.11p we aim to compensate the cons of the vision sensor systems which are mentioned above. Due to importance of real time requirement, we also made tests on the GPU accelerated NVIDIA Tegra Jetson TK1 development board and compare with the CPU results for future development.

RÉSUMÉ

De nos jours, les voitures autonomes commencent déjà à être utilisées dans notre vie quotidienne. Ce sont les véhicules de nouvelle génération qui sont capables de fonctionner à certaines conditions sans avoir besoin d'une réaction du conducteur. Ils peuvent suivre les marqueurs de la voie et se maintenir à l'intérieur de celle-ci, localiser les véhicules autour et décider et effectuer un arrêt ou même changer la voie actuelle en cas de nécessité. Les systèmes d'aide au conducteur représentent la technologie sous-jacente de ces véhicules. Ils permettent aux véhicules d'analyser les conditions routières et la circulation afin de prendre ce type d'actions nécessaires. Cela rend la tâche de la détection des objets sur la route tels que les véhicules, les marqueurs de la voie, les rambardes et même les piétons très importante. Pour cette raison, il est également très important de pouvoir fonctionner en temps réel.

Les études actuelles portent principalement sur le capteur de vision, le LIDAR, les capteurs micro-ondes, etc. Cependant, seuls, ils souffrent de problèmes tels que la grande variété de cibles, les fluctuations d'éclairage, la courte portée, la nécessité d'une haute puissance d'opération, les coûts etc. Dans cette étude, nous introduisons un système à base de caméra monoculaire, qui est amélioré selon la norme IEEE 802.11p concernant la communication de véhicule-à-véhicule. A l'aide de la norme IEEE 802.11p, nous visons à compenser les inconvénients des systèmes de capteurs de vision qui sont mentionnés ci-dessus. En raison de l'importance des exigences en temps réel, nous avons également effectué des tests sur le GPU accéléré NVIDIA Tegra Jetson TK1 carte de développement et comparé avec les résultats du CPU pour le futur développement.

ÖZET

Günümüzde otonom araçlar trafikte aktif olarak kullanılmaya başlanmış durumda. Bu araçları sürücüden geri bildirim almaksızın belirli koşullarda kendini sürebilen taşıtlar olarak tanımlayabiliriz. Otonom araçlar şeritleri algılayıp takip edebildiği gibi diğer araçları da algılayarak gerekli durumlarda fren sistemini devreye sokabilmekte ve hatta şerit değiştirebilmektedir. Bu teknolojinin alt yapısı ise sürücü destek sistemleri tarafından oluşturulmaktadır. Bu sistem sayesinde araçlar yol ve trafik durumunu analiz ederek gerekli eylemleri uygulayabilmektedirler. Bu işlemler yol üzerindeki diğer araçların, şeritlerin, bariyerlerin ve hatta yayaların algılanması problemini önemli kılmaktadır. Bu nedenle bu sistemlerin gerçek zamanlı olarak çalışma gereksinimi de doğmaktadır.

Bu alanda yapılan çalışmalar çoğunlukla görüntü sensörü, LIDAR, mikrodalga sensör vb. kullanımı ile gerçekleştirilmiştir. Ancak bu cihazlar hali hazırda nesnelerin çok çeşitlenmesi, ışık değişimleri, kısa menzil, yüksek işlem gücü gereksinimi ve maliyet gibi parametrelerden etkilenmektedir. Bu çalışmamızda IEEE 802.11p araç-araç haberleşmesiyle desteklenmiş tekil görüntü sensörü tabanlı bir sistem önerilmektedir. IEEE 802.11p araç-araç haberleşmesi sayesinde yukarıda belirtilen görüntü sensörünün eksilerinin telafi edilmesi hedeflenmiştir. Ayrıca gerçek zamanlı çalışma gereksiniminden dolayı NVIDIA Tegra Jetson TK1 geliştirme kartı kullanılarak üzerindeki GPU üzerinde de performans testleri yapılmış ve CPU sonuçları ile de karşılaştırılmıştır.

1. INTRODUCTION

Driver assistance systems are the important part of autonomous cars of future. It is the background technology of self-driving cars which are designed to replicate the human behavior. Even today it is already possible to see these vehicles at the streets. The vehicle can drive itself without the input of a driver, keep itself inside the lane, keep the necessary distance from the front car and even change the present lane when the required conditions met. In order to achieve these tasks, the detection of other objects such as lane markers, vehicles, traffic signs in the environment is very essential. For instance, in case of pre-crash detection, one has to detect the front vehicle and estimate the distance as well as the speed of that vehicle to take necessary actions when it took a sudden break in order to prevent any possible fatal accident. Such a system needs to be robust and real time to take these necessary actions immediately. For this reason, accurate detection of other objects around the vehicle is a highly important task.

Current studies at this area can be based on vision sensors, microwave sensors and LIDAR. However, they each have limited performance and accuracy. For instance, detection with standard vision sensors would be inaccurate at longer distances due to reduced resolution, in scenes with bad weather conditions as well as the sudden changes in lighting. Also, the variety of the targets is another problem at this task. Vehicles may come with different shapes and sizes. Other than that, LIDARs are still very expensive products and they need high computational power while boards deployed in vehicles are tend to be small and cost effective. Microwave radars are much more cost effective than LIDARs, however they suffer from reflections from concrete objects on the road. On the other hand, our approach motivated by the vision sensor and IEEE 802.11p V2V communication, targets these downsides for a better detection results without the need of a high computational power.

For the real time performance requirement, we also test the algorithms in a GPU accelerated board as well as today's regular CPU. We used NVIDIA Tegra Jetson TK1 GPU accelerated board which has the mobile processor to have the same advanced features & architecture as a modern desktop GPU while still using the low power draw of a mobile chip and therefore it's very suitable for these kinds of applications.

Systems like these need to be tested very well on the real world conditions. Otherwise its performance and importance might never be evaluated and cannot be used in such critical scenarios. We also tested our system in real time conditions such as rural roads, highways and we will be able to get live results to evaluate our system's performance in such conditions.

2. LITERATURE REVIEW

Object detection has been widely studied in many applications for years. For instance, authors focused on vehicle detection based on radar and vision fusion to automatically activate the car's emergency braking system (Kim & Song, 2013). They asserted that the radar cannot successfully recognize if the detected object is a real vehicle or not. Although the radar's performance in radial direction is high, they give its coarse performance in azimuth direction. This problem causes false detection of a preceding vehicle in the same lane, resulting in false activation of automatic emergency braking. And to improve this false detection, they suggest a vehicle recognition method based on the shape and motion in which the motion attribute is to determine whether the object is either stationary or dynamic and the shape attribute is to identify whether the objective is a real vehicle or not. Figure 2.1 shows the flowchart of this algorithm. They also supported their algorithm with field test data.

Çayır and Acarman (2009) have built a low cost driver monitoring and warning system to warn the driver when the car leaves the lane by the help of a single camera. Like so, in a recent work of Huo et al. (2012), another lane departure warning system has been studied to assist the driver when the car exits the lane. They used a radar which uses a Doppler Effect to detect the frequency shift in reflected waves and vision-based camera to recognize the patterns on the road such as lane-marking, front vehicle, road sign and other obstacles. This is not a vehicle detection based application, however the authors have targeted the real time lane detection and tracking which is another important task in active driving assistance systems in ITS.

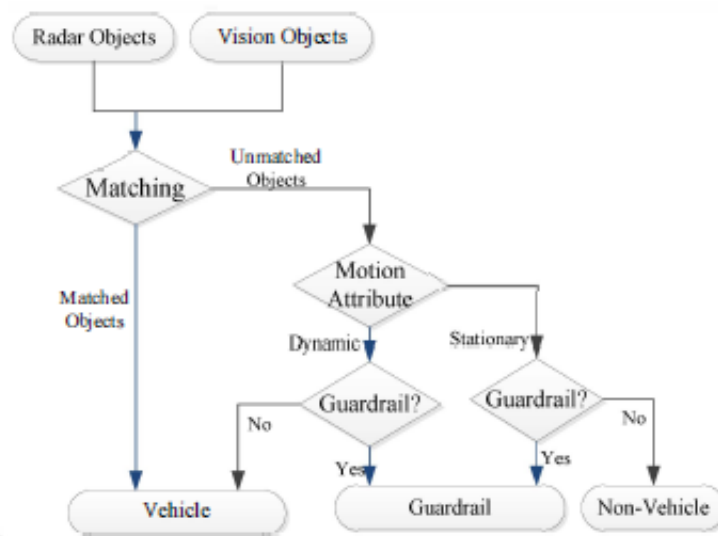


Figure 2.1: Vehicle Validation (Kim & Song, 2013)

However, Alessandretti et al. (2007) introduced a vehicle detection system which was a fusion of radar and vision data. Additionally, they included the guard rail detection and also a method to handle the overlapping areas. They used radar data to locate the areas of interest on the images to perform the detection task. Vehicles found in different image regions were combined together and series of filters were applied to remove the false detections. They found out that their systems performed good results in both rural and highway environments. They didn't detect all the vehicles in all images, however their system was promising enough for driver assistance applications since the closest and most dangerous vehicles are correctly and precisely detected. Table 2.1 shows their system's execution time performance. From the figures in Table 2.1 it is possible to say that the more guard rails are detected, the more time is saved. Specifically, in sequence 12, many guard rails were present and detected, and the time saved is about 36%. They also state that a reduction of 4-5 ms in execution time is very significant in such real time applications.

Another real time application was proposed by Betke et al. (2000) in order to develop an intelligent, camera-assisted car which is able to interpret its surroundings in real time basis. The system that has been developed at this work, analyzes color videos which are

grabbed from a forward-looking video camera mounted in a car. They used a combination of color, edge and motion information to detect the vehicles, road boundaries and lane markings on the road. Additionally, they present a method for detecting cars that is a temporal differencing and tracking motion parameters that are typical for cars. They evaluated their system in American and German highways both during both the day and night. They achieved reasonably good result even with a low-cost PC and an image capture board. Table 2.2 shows the processing times of this work.

Table 2.1: Execution time on highways. VD stands for vehicle detection algorithm only, GRD stands for guard rail and vehicle detections algorithms and OBM stands for overlapping boxes management, guard rail and vehicle detection. The decrement of OBM with respect to VD is shown in the reduction column (Alessandretti et al., 2007)

sequence	VD [ms]	GRD [ms]	OBM [ms]	reduction
11	22	19	17	22%
12	25	18	16	36%
13	30	28	26	13%
14	27	24	22	18%
15	29	27	26	10%
average	26,6	23,2	21,4	20%

Bertozi et al. (1998) studied the stereo vision based obstacle and lane detection on moving vehicles in order to increase the road safety. They focused on detecting both generic obstacles and the lane positions at a rate of 10 Hz. They needed to use a custom massively parallel hardware for this work. By the help of this specific hardware perspective effect from stereo images was removed. They used left vision sensor to detect lane marking while remapped stereo images were used to detect the free space in front of the vehicle. Rezaei et al. (2015) has studied another real time monocular camera based application for vehicle detection and as well as inter-vehicle distance estimation. They used Haar based features and pixel based distance estimation.

Table 2.2: Average processing time of each algorithm (Betke et al., 2000).

Step	Time	Average time
Searching for potential cars	1-32 ms	14 ms
Feature search in window	2-22 ms	13 ms
Obtaining template	1-4 ms	2 ms
Template match	2-34 ms	7 ms
Lane detection	15-23 ms	18 ms

In a recent work of Yıldız and Acarman (2012) an extended SIFT feature description method has been studied in order to achieve robust vehicle tracking. Sun et al. (2006) focused on a pre-crash vehicle detection system as well and evaluated their work with different feature extraction methods as well as classifiers. As a result, they achieved an average detection rate of 10 Hz. The data was collected and processed in real time on a moving car while cruising under different traffic conditions. Their algorithm consisted of two parts: A hypothesis generation (HG) step in which the image location where the vehicles might be present were extracted and the verification (HV) step which verifies the hypothesis. They used PCA, Gabor and Wavelet feature extraction methods. They found out that regardless of the feature extraction method each time Support Vector Machine gave better results on vehicle detection in comparison to Neural Networks. Figure 2.2 shows the result of the system when Wavelet features used with both SVM and Neural Networks. 15, 20, 25, 30 and 35 are the hidden nodes used in neural network.

They found out that using SVM (referred as WS), the average error was 8.52%, the average FP rate was 6.50% and the average FN rate was 2.02%. Next they evaluated the performance of wavelet features using NN, referred to as WN in Figure 2.3. The lowest error measured as 16.4% (FP 12.81% and FN 3.59% and was achieved by a NN with 30 hidden nodes. SVM performed better than NN using wavelet features).

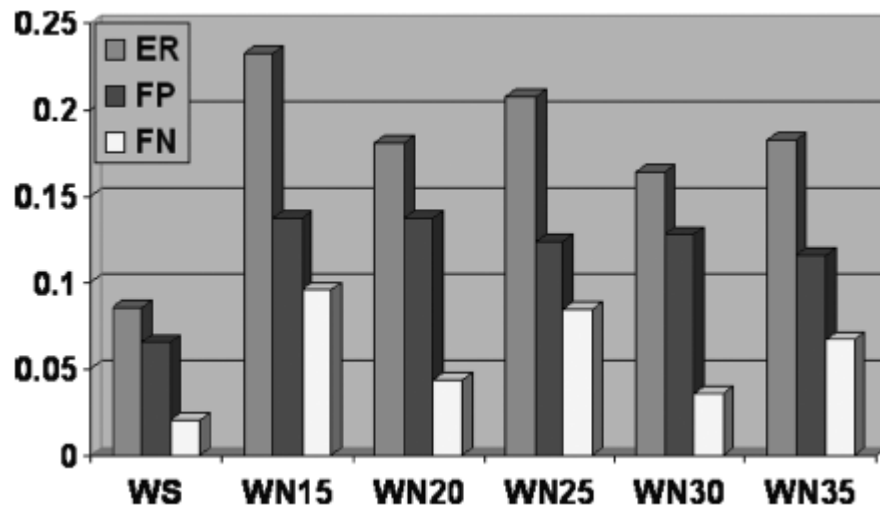


Figure 2.2: HV using wavelet features (Sun et al. 2006)

There have been other authors like Caraffi C. et al. (2012), Jazayeri A. et al. (2011), Chen Y. et al. (2011) who aimed at similar tasks such as safety, auto-driving, driver assistance systems for ITS which are based on visual sensors and as well as the autonomous driving system which uses six LIDARs in order to detect pedestrian crossing, speed-bump and obstacles on road by Choi J. et al. (2012).

3. MATERIALS AND METHODS

In this work we focused on robust and reliable vehicle and lane marker detection on a moving vehicle at real time conditions as well as distance and relative and ground speed estimation. We propose a system, which detects the vehicles simultaneously at real time conditions by the help of a standard visual sensor. Also we support our monocular visual sensor based system with a V2V (Vehicle-to-vehicle communication) on-board unit.

Direct communication between the vehicles allows information exchange without requiring any fixed infrastructure or base stations. The location and velocity of vehicles are constantly changing and the RF communication range is fairly short distance; therefore, the set of vehicles that can directly communicate will constantly change over a short period of time. This dictates that the physical layer and the network must be capable of operating in an ad hoc, decentralized manner, although coordination and synchronization through GPS time signals are possible. Any two nodes must be able to communicate securely whenever they are within the communication range.

In a V2V network we can distinguish two modes of communication, usually designated as:

- **Single hop:** Two vehicles are close enough to communicate directly with each other (either broadcast or point to point) with low latency.
- **Multi hop:** Vehicles that cannot directly communicate may forward messages through intermediate nodes.

Multi hop communication has been the subject of much research (Korkmaz et al. 2006), but no standard has emerged, and in fact the technical difficulties of establishing routing and acknowledgment protocols along with potentially high latency may limit its use to very specific applications such as medium range emergency notification or other sparse broadcast communication applications.

Many early experiments in V2V communication were carried out with standard wireless LAN, for example IEEE 802.11b, operating in the 2.4-GHz ISM band, and some success was achieved at ranges up to several hundred meters. But the technical difficulties inherent in vehicle and traffic situations, including the high relative velocities (Doppler effects), a safety critical low latency requirement, operation in an urban environment (multipath), and spectrum competition from other users in unlicensed frequency bands renders this an unrealistic solution for commercial deployment. The IEEE 802.11p/WAVE standards have recently emerged as the current consensus for implementation of V2V and local V2I communications.

Dedicated Short-Range Communication (DSRC) systems are short- to medium-range communications systems intended to cover communication ranges of 10–1,000m. The term DSRC has come to refer to automotive or mobile applications. A number of technologies have been identified, but the current object of worldwide standardization activities are variants of the 802.11p/WAVE standard operating in the 5.9-GHz range. The United States has currently allocated 75 MHz of spectrum for DSRC applications, and the EU has allocated 35 MHz of overlapping spectrum.

Other DSRC technologies include a 902–928-MHz band standard (ASTM E2158-01) that has primarily been used in electronic toll collection and commercial vehicle operation applications. It is incompatible with the 5.9-GHz DSRC standards.

A Japanese industrial consortium, including OKI Electronics Ltd., developed the “Dedicated Omnipurpose Intervehicle Communications Linkage Protocol for Highway Automation” (DOLPHIN) (Tokuda et al. 2000, Shiraki et al. 2001, Tsugawa et al. 2001) system operating in the 5.8-GHz band which provided broadcast, point-to-point, and

broadcast with packet forwarding capabilities. Bluetooth, various UWB, WiMAX, and even Zigbee (IEEE 802.15.4) could also be considered for DSRC applications.

In order to comply with the European Profile Standard ITS-5G defined in ETSI EN 302 571, IEEE 802.11p physical and MAC layers are implemented. Cooperative Awareness Messages (CAM) broadcast with a value of 19 dBm on the 10 Mhz channel at 5.9 GHz with a data rate of 3 Mbit/s. List of parameters can be seen in Table 3.1.

Table 3.1: List of parameters in IEEE 802.11p

Parameter	Value
Scenario	Urban Canyon
Vehicle Velocity	Max 20 m/s
Transmit Rate	3 Mbps
Transmit Power	19 dBm
Carrier Sensing Threshold	-99 dBm
Transmit Range	550 m
CAM Rate	1 Hz

We used C++ as the programming language of our system and took advantage of OpenCV image processing library. OpenCV provides us useful algorithms which can both run in CPU and as well as in GPU with only few refinements. We also started to implement important parts of our system both into the CPU and GPU architecture. We made several tests and compared the CPU and GPU performances of individual algorithms.

3.1. Vehicle Detection

We used Haar feature-based cascade classifier for detection. This object detector has been initially proposed by Viola P. and Jones M. (2001) and later its features and efficiency improved by Lienhart R. et al. (2003). Haar classifier is trained with a few

hundred samples of target object which are called positive examples and non-target examples are called negatives. After the classifier is trained, it features several classification stages which are applied to a ROI within the image. Table 3.1 shows the features used in the algorithm. For weighting, pixel sum difference between black areas and white areas calculated by using integral image (shown in equation 1 and Figure 3.1) that is used for fast feature evaluation.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (1)$$

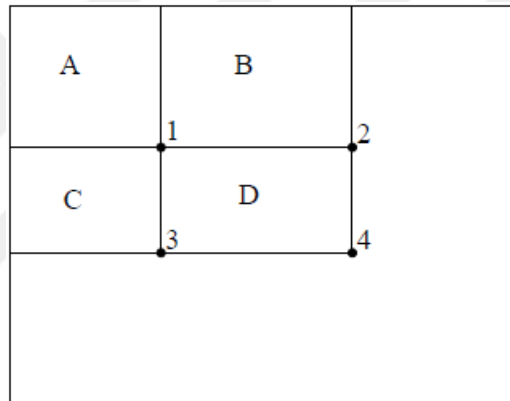


Figure 3.1: Integral image method. Sum of pixel values in D region can be calculated as $4+1-(2+3)$.

We then trained our own vehicle classifier as a starting point. We used *UIUC Image Database for Car Detection* for training. This database provides 550 positive and 500 negative samples collected at the University of Illinois. The resolution of each positive and negative training images are 100x40 pixels. Figure 3.2 shows sample images from this database.

Training 550 positive and 500 negative samples took more than 3 hours. Training was finished in a test pc running a 64 bit Linux Ubuntu operating system. Figure 3.3 shows

the output of the trained system. It can be seen that the car in the test image has been detected successfully and circled in white.

We also made real time tests at test area of my company. Figure 3.4 shows the performance of the current system. Resolution of the live images which are retrieved from the camera is 640x480 pixels.

Table 3.2: Used Haar features from Lienhart R. et al (2001). Black regions have negative and white regions have positive weights.

Edge features								
Line features								
Center-surround features								

We tested our system's performance from image sequences data which are collected in real time on our test car while cruising. We ran tests both on city traffic as well as highways to understand the behavior of our system at different environments. We prepared a simple interface showing real time camera frames and marked refined vehicle positions. According to the Figure 3.4, it is possible to say that current system's detection rate is about 14-15 Hz even though we performed the detection algorithm at each retrieved frame. During the tests camera was set to stream at a rate of 20 fps.



Figure 3.2: Sample images from UIUC image database. First row shows negative, second row shows positive and the third row shows the test images

```

mustafa@mustafa-VirtualBox: ~/build-CarDetect-Desktop_Qt_5_4_0_GCC_64bit-Debug
mustafa@mustafa-VirtualBox:~/Desktop$ cd ..
mustafa@mustafa-VirtualBox:~$ pwd
/home/mustafa
mustafa@mustafa-VirtualBox:~$
mustafa@mustafa-VirtualBox:~$
mustafa@mustafa-VirtualBox:~$
mustafa@mustafa-VirtualBox:~$
mustafa@mustafa-VirtualBox:~$
mustafa@mustafa-VirtualBox:~$ cd build-card
bash: cd: build-card: No such file or directory
mustafa@mustafa-VirtualBox:~$ cd build-CarDetect-Desktop_Qt_5_4_0_GCC_64bit-Debug/
mustafa@mustafa-VirtualBox:~/build-CarDetect-Desktop_Qt_5_4_0_GCC_64bit-Debug$ ls -l
total 544
-rwxrwxr-x 1 mustafa mustafa 201239 Mar 31 19:55 CarDetect
-rw-rw-r-- 1 mustafa mustafa 323104 Mar 31 19:55 main.o
-rw-rw-r-- 1 mustafa mustafa 27189 Mar 31 19:46 Makefile
mustafa@mustafa-VirtualBox:~/build-CarDetect-Desktop_Qt_5_4_0_GCC_64bit-Debug$
mustafa@mustafa-VirtualBox:~/build-CarDetect-Desktop_Qt_5_4_0_GCC_64bit-Debug$
mustafa@mustafa-VirtualBox:~/build-CarDetect-Desktop_Qt_5_4_0_GCC_64bit-Debug$
mustafa@mustafa-VirtualBox:~/build-CarDetect-Desktop_Qt_5_4_0_GCC_64bit-Debug$ .
/CarDetect

```

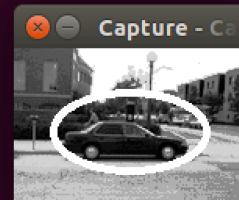
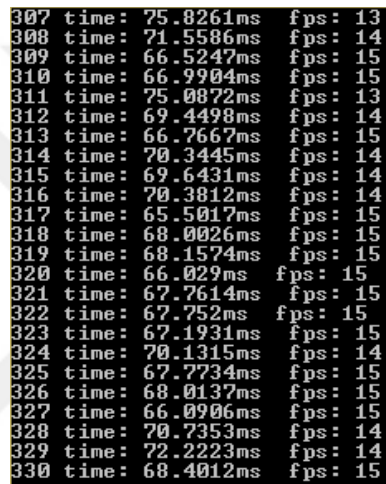


Figure 3.3: Detected car on test image of UIUC image database

After that we started to retrieve each frame collected from the road in order to analyze the road for vehicle and lane marker detection. The lane marker detection is important for evaluating our car's position inside the lane. We first applied smoothing to each frame to get rid of the noises and then used Canny edge detection algorithm to extract edges which was introduced by Canny J. (1986). Afterwards, we applied Hough Line Transform by Matas J. et al., (2000) for possible lines on the road. Figure 3.5 shows the output of this stage.



```

307 time: 75.8261ms fps: 13
308 time: 71.5586ms fps: 14
309 time: 66.5247ms fps: 15
310 time: 66.9904ms fps: 15
311 time: 75.0872ms fps: 13
312 time: 69.4498ms fps: 14
313 time: 66.7667ms fps: 15
314 time: 70.3445ms fps: 14
315 time: 69.6431ms fps: 14
316 time: 70.3812ms fps: 14
317 time: 65.5017ms fps: 15
318 time: 68.0026ms fps: 15
319 time: 68.1574ms fps: 15
320 time: 66.029ms fps: 15
321 time: 67.7614ms fps: 15
322 time: 67.752ms fps: 15
323 time: 67.1931ms fps: 15
324 time: 70.1315ms fps: 14
325 time: 67.7734ms fps: 15
326 time: 68.0137ms fps: 15
327 time: 66.0906ms fps: 15
328 time: 70.7353ms fps: 14
329 time: 72.2223ms fps: 14
330 time: 68.4012ms fps: 15

```

Figure 3.4: Detection time of cars in our trained system

We use this information to estimate the lane borders and draw them on the screen so if the car does not cruise at the center of the lane our system will inform the driver. After that we applied the car detection task by using our generated trained HAAR classifier before. We saw that our system is able to detect the front vehicles at close distances.

3.2. Distance and Relative Speed Estimation

We used pixel based inter-vehicle distance estimation (Rezaei et al. 2015) in order to calculate the front cruising vehicle's distance to the ego vehicle as in Figure 3.6. We also estimated relative speed of the cruising vehicle in front from the distance data. We changed the proposed algorithm of Razeai et al. The proposed algorithm requires the

vision sensor to be mounted at a tilt angle so that the projection of lower bound of image must exactly correspond to the distance between the front of the car and camera.



Figure 3.5: Image analysis for the lane marker detection

However, with this installation hood of the car is quite visible and taken into account in algorithms like edge and feature detection. For this reason, we decreased the tilt angle so that only road is visible at lower part of the image. As a result, our formula for distance estimation changed as:

$$d2 = \tan(\theta_c \pm \beta) \times H \quad (2)$$

$$D = d_2 - d_1 \quad (3)$$

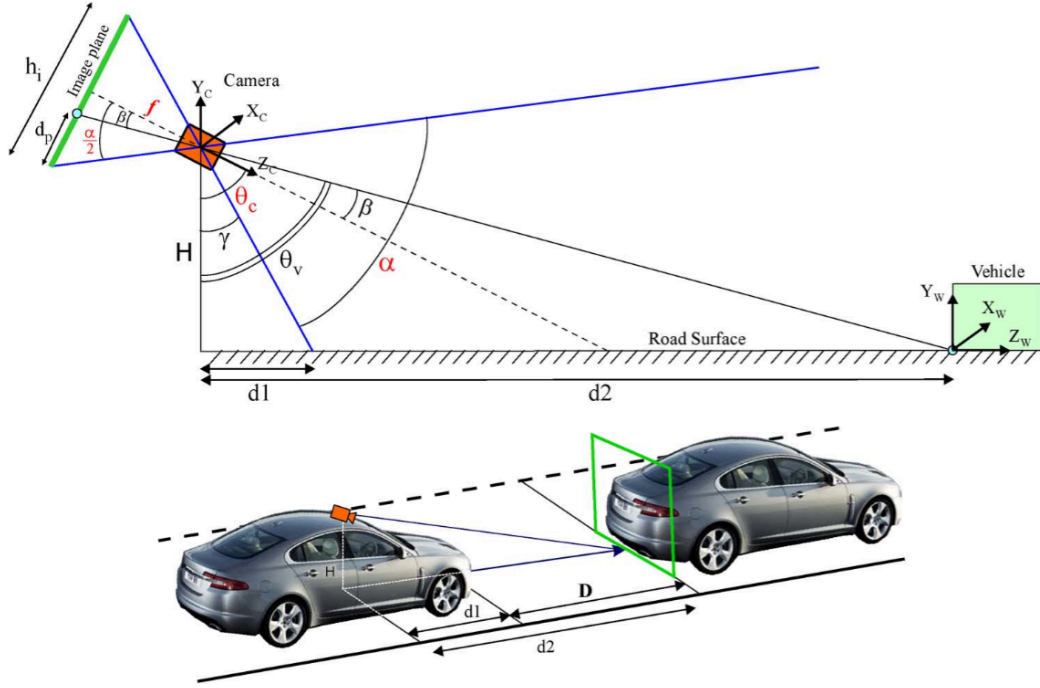


Figure 3.6: Pixel based inter-vehicle distance estimation in 2D image plane (Rezaei et al. 2015)

where H stands for the camera height, θ_c is the angle between the Z_c and $-Y_c$ axis, d_1 is the length from camera to the front of the vehicle, d_2 is the distance between camera and the front vehicle and D is the distance between front vehicle and the ego vehicle. One parameter in this calculation which is not explained clearly is the field of view (FOV) of the camera. FOV of the camera depends on different parameters such as focal length of the lens, sensor width and height. Using these parameters, one can calculate the FOV as:

$$\alpha = 2 \times \tan^{-1} \left(\frac{H/2}{f} \right) \quad (4)$$

$$\beta = 2 \times \tan^{-1} \left(\frac{W/2}{f} \right) \quad (5)$$

where H stands for the sensor height in mm, W is the sensor width in mm, f as the focal length in mm, α is the vertical FOV and β is the horizontal FOV. The sensor information can be found in manufacturer's datasheet.

After distance estimation is finished we calculated relative speed of the vehicle based on the distance travelled as in (6).

$$V_r = 3600 \times \frac{\pm \Delta_d}{1000/H} \quad (6)$$

V_r stands for relative speed of the detected vehicle in kmph, Δ_d is the distance in meters that the vehicle travelled between two consecutive frames and H is the frame rate of the camera.

3.3. Tracking

We made tests with different trackers such as MIL, BOOSTING, TLD, KCF. However, we saw that MEDIAN FLOW tracker introduced by Kalal et al. (2010) showed better results in our scenario, specifically in vehicle tracking. The algorithm is based on failure detection that is measured by the differences due to the occlusions between video frames. Such an occlusion can be seen in Figure 3.7.

Given the pair of images and a bounding box (vehicle ROI in our case), it generates a set of points inside box and applies Lucas-Kanade tracker which computes a sparse motion flow between frames. According to the quality of the point predictions each point is assigned an error, before filtering out the 50% of the worst errors as outliers. The remaining points are used to estimate the displacement of the whole bounding box. As a result, bounding box gets updated as the object moves in time. Figure 3.8 shows the process of the algorithm.

Each of the tracked boxes are either given an incremental unique id or current id is maintained through the lifecycle of the vehicle. Algorithm of our detection and tracking application can be seen in Figure 3.9. When the new image is received from the camera we preprocess the image by converting it to grayscale and apply blurring. The image is then given as an input to detection algorithm to generate vehicle hypotheses as well as lane markers. Each hypothesis is then verified by whether its representing the line information as the vehicle forms. We simply counted horizontal lines that can form a vehicle to verify our candidates rapidly.

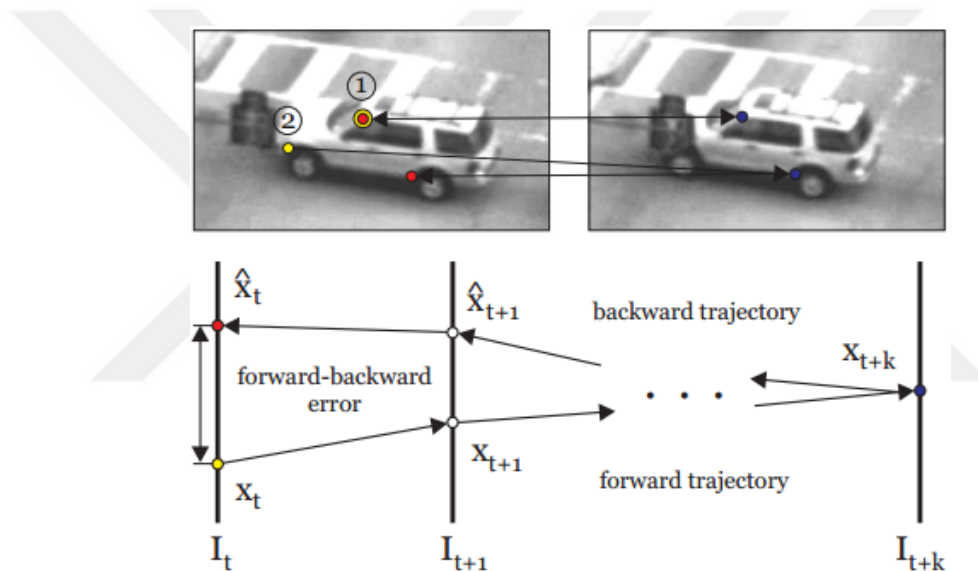


Figure 3.7: The Forward-Backward error (Kalal et al. (2010)). Point 2 is occluded in the next frame and tracker matched a different point. Tracking back from this point to the previous frame ends in a different location which ensures that the matched point is inconsistent.

As the next step we gave each detection to our tracker as an input. However, our tracking algorithm firstly checks whether the given vehicle whether is already in tracked state before initializing the new window. If so, our tracker simply ignores or otherwise, it assigns this new window a unique id, performs Forward-Backward error estimation and updates its location at each frame until its invisible.

3.4. Sensor Fusion

So far, we could only have a rough estimation of relative speed of other vehicles. For ground speed estimation, we needed to know our speed as well. Thus, we could estimate the speed of the target vehicle in reference to the ground. However, we also want to enhance our camera measurement with another sensor that is IEEE 802.11p V2V communication in our case. For this reason, our GPS communication software run in our IEEE 802.11p unit. For our offline tests, we needed real road data so we can evaluate our system's performance and outputs. We covered this topic in the next chapter.

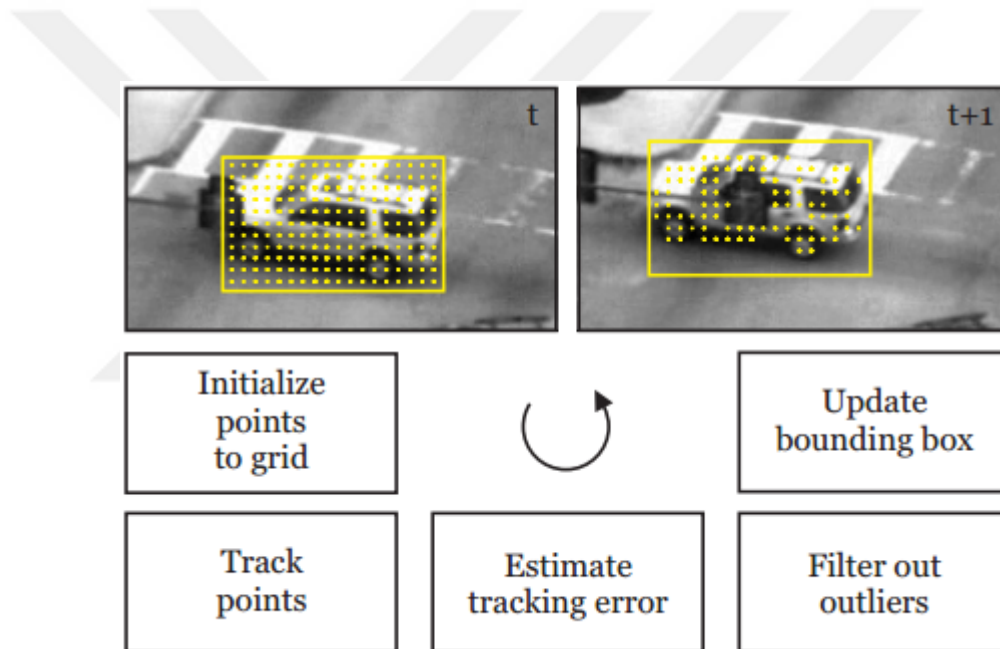


Figure 3.8: The Median Flow algorithm by Kalal et al. (2010)

After collecting real road measurements with our camera and IEEE 802.11p units we had information of a particular vehicle's position and speed from two different sensors. Using our detection method, the camera gives us the pixel based position information in 2d which is later used for estimating the local coordinates X and Y in meters as well as the distance and relative speed in kmph. However, from our IEEE 802.11p communication we have the geographic coordinate information that is latitude, longitude and altitude of target and ego vehicle. In order to support our camera measurement, we needed the

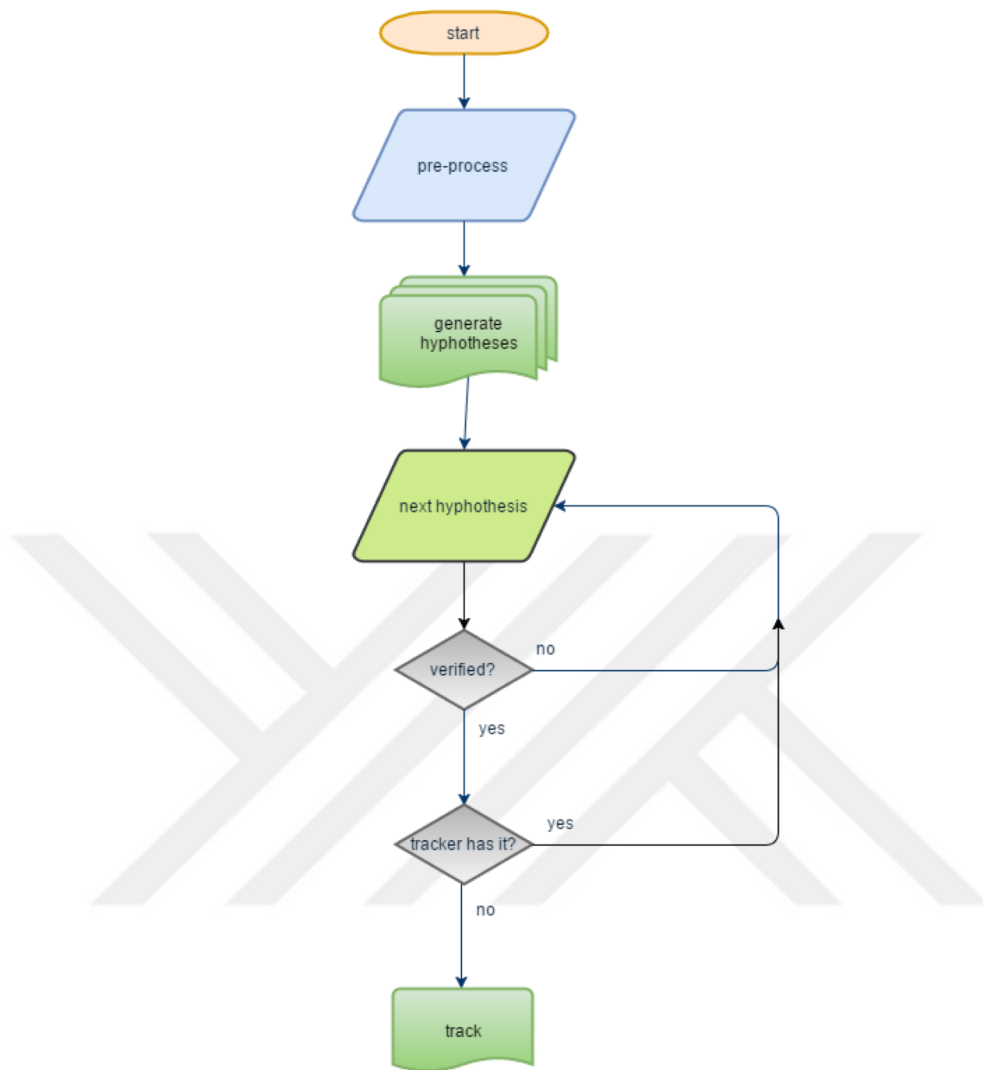


Figure 3.9: Detection and tracking algorithm of our application

position information in the same units as we have both speed information in kmph. To achieve this, firstly we converted each latitude, longitude and altitude information from the detected vehicles as well as ego vehicle to the global earth-centered, earth-fixed (ECEF) coordinates in meters. Next, the difference in ECEF coordinates between the detected vehicle and ego vehicle are converted to east, north, up (ENU) local coordinates. Since we converted the difference of ECEF coordinates, we had the difference ENU coordinates which provide us east, north and up difference in meter between the detected vehicle and ego vehicle. However, this result gives us the ENU coordinate difference in

reference to east, north and up axes. We need the X, Y and distance information towards the detected vehicle with our current heading. Using the rotation matrix with our heading information and local ENU position vector we obtained the local coordinates to the detected vehicle (7). Here we assume that the road is flat and we ignore the *up* figure.

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \times \begin{bmatrix} e \\ n \end{bmatrix} \quad (7)$$

After finding calculating the local coordinates of the detected vehicle we calculated its Euclidean distance. In this calculation, we have taken into account the distances between bumpers of the vehicles and the GPS receiver as explained in Figure 4.1.

We used the particle filter algorithm in order to fuse our sensor measurements. We used 200 particles, used 3 meters and 4 meters as measurement variance for our vision sensor and V2V sensor, respectively. Particle filter is a numerical approximation to the Bayes Filters. The sampling importance resampling (SIR) algorithm is one of the most widely used sequential Monte Carlo methods, which allow the system state estimation to be computed on-line while the state changes as it is the case for tracking algorithms. A SIR filter usually manages a fixed number of possible system state hypotheses x_t^i , where superscript i denotes the i -th individual particle. These individual particles approximately generate the distribution of the system state, $p(X_t)$. The SIR algorithm is computed at each discrete time step. Algorithm of such system can be seen in Figure 3.10.

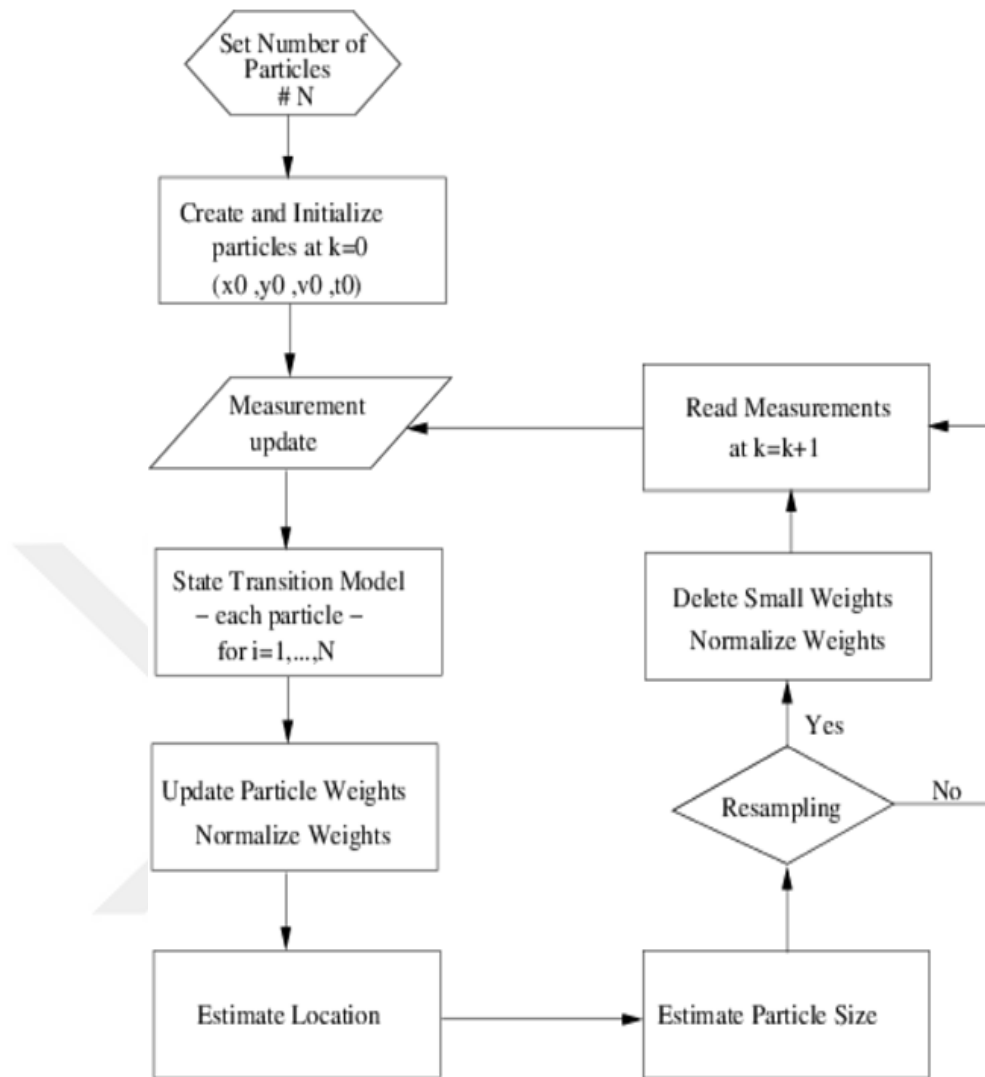


Figure 3.10: Particle filter algorithm

4. EXPERIMENTAL STUDY

For our system's evaluation, we also collected data at both highway and in-city roads of Istanbul, Turkey. We used two cars to collect data. One of them has the camera sensor, IMU, IEEE 802.11p unit and GPS receiver on it, which we will be calling CAR A. The other car has IEEE 802.11p unit and the GPS receiver, which we will be calling CAR B. An illustration of such system can be seen in Figure 4.1. We used a 1080p network camera with 8 mm focal length in CAR A.

We prepared a data collector and a sender software for our system. The data collector runs in a laptop in CAR A and stores the data received from IEEE 802.11p, GPS, IMU as well as live frames from the camera. The data sender runs on IEEE 802.11p unit also in CAR A and sends the received CAM through TCP communication to our data collector. The schema of this communication can be seen in Figure 4.2. The example of the received dataset can be seen in Table 4.1 and 4.2.

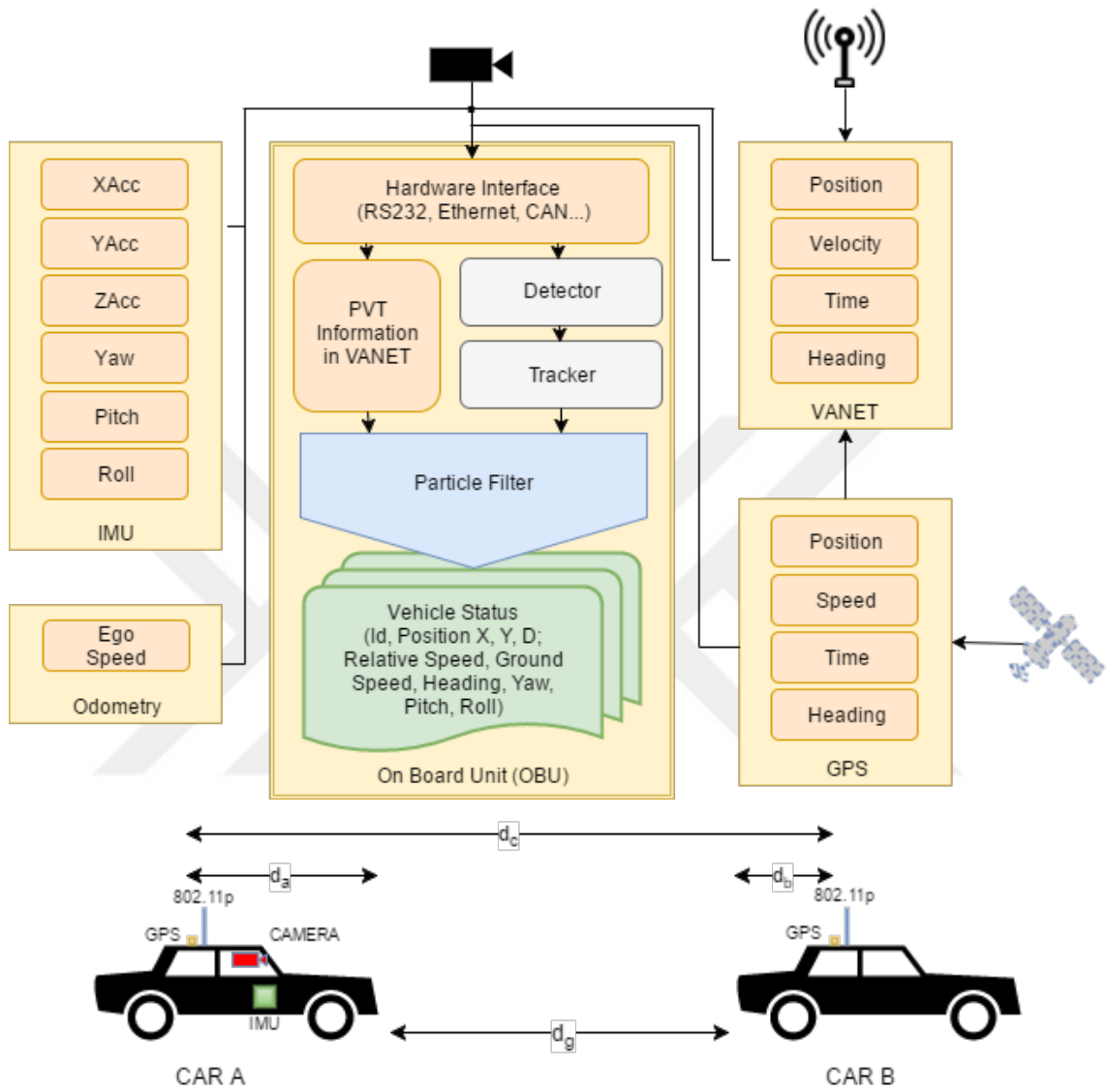


Figure 4.1: Test system used for data collecting. d_a , d_b , d_c , and d_g refers to the distance from GPS receiver to the front bumper in CAR A, distance from GPS receiver to the rear bumper in CAR B, estimated distance between GPS receivers of the two cars and actual distance between the cars, respectively.

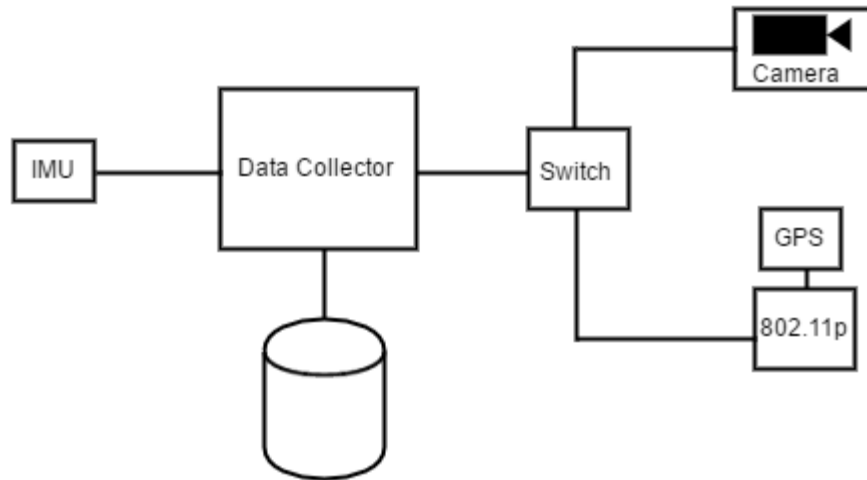


Figure 4.2: Data collection method.

In Table 4.1 the samples from CAM data are showed. Every message consists of two sets of variables, which are latitude, longitude, altitude in meters, heading angle, speed in km/h. The first set is gathered from the ego vehicle CAR A while the second set is from the IEEE 802.11p unit in CAR B. Each of the data is associated to a frame number that corresponds to the last frame retrieved from the camera. One important note here is that our camera was set to operate at 20 fps during the data collection.

Table 4.2 shows the IMU samples received from our IMU unit. The first 3 figures are the acceleration in X, Y and Z axes where the last 3 figures are the angular rates in X, Y and Z axes. IMU measurements are also associated with the corresponding frame number. Since our IMU device much faster than our camera sensor, we were able to store 1 measurement for each camera frame unlike in IEEE 802.11p communication. The device orientation during the data collection can be seen in Figure 4.3.

Table 4.1: CAM samples received from the data sender in IEEE 802.11p.

Frame No	18350	18352	18353	18355	18358	18370	18374	18388	18418
Lat1	41.045 2701	41.045 3373	41.045 3373	41.045 3373	41.045 3373	41.045 4036	41.045 4036	41.045 4036	41.045 5236
Lon1	29.017 8846	29.018 0603	29.018 0603	29.018 0603	29.018 0603	29.018 2310	29.018 2310	29.018 3938	29.018 5480
Alt1	6.90	7.09	7.09	7.09	7.09	7.59	7.59	8.09	8.19
Heading1	63.0	63.2	63.2	63.2	63.2	63.0	63.0	63.5	63.2
Speed1	60.18	58.92	58.92	58.92	58.92	57.27	57.27	53.86	51.27
Lat2	41.045 5274	41.045 5869	41.045 5869	41.045 5869	41.045 5869	41.045 6373	41.045 6373	41.045 6831	41.045 7198
Lon2	29.018 5476	29.018 6916	29.018 6916	29.018 6916	29.018 6916	29.018 8261	29.018 8261	29.018 9478	29.019 0550
Alt2	11.69	12.30	12.30	12.30	12.30	12.50	12.50	12.50	12.59
Heading2	62.3	61.9	61.9	61.9	61.9	64.1	64.1	63.6	65.3
Speed2	50.80	50.80	49.16	49.16	49.16	49.16	44.45	44.45	34.01

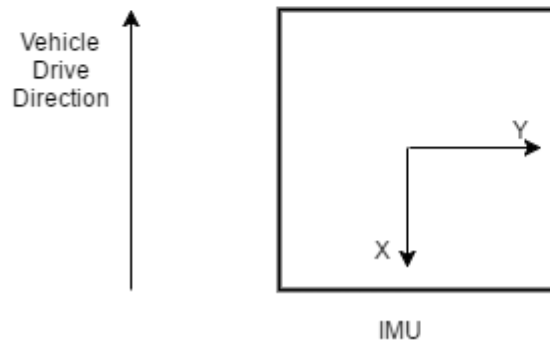


Figure 4.3: IMU unit orientation and vehicle drive direction.

Table 4.2: IMU samples.

Frame No	AccX	AccY	AccZ	RateX	RateY	RateZ
18350	-0.0595093	0.0430298	-1.06232	-0.480652	2.48016	0.0192261
18351	-0.0695801	0.0479126	-1.01013	-1.59576	0.596008	-0.115356
18352	-0.0457764	0.00274658	-1.02356	-0.211487	1.07666	0
18353	-0.0543213	0.0683594	-1.10443	-1.24969	0.480652	0.211487
18354	-0.0387573	0.0534058	-0.844421	0.499878	0.365295	0.115356
18355	-0.0283813	0.06073	-0.987549	1.3266	-1.90338	-0.0769043
18356	-0.0280762	0.100098	-0.974426	-0.0192261	-2.7301	-0.0576782

We used real traffic data collected in Istanbul, Turkey as explained. We ran our algorithms with this collected data and analyzed the results. However, we also compared hit rate of our classifier with well-known vision benchmark of KITTI from Geiger et al. The samples from this data set can be seen in Figure 4.4 and the results of the comparison can be seen in Table 4.3. One can see that we received better precision performance with our road data. We believe that better quality of our vision sensor data caused this difference.



Figure 4.4: Sample from KITTI vision benchmark.

Table 4.3: Comparison of TPR (Recall), PPV (Precision) of KITTI vision benchmark and our road data from Istanbul.

Dataset	TP	FP	FN	TPR	PPV
KITTI	26	9	6	81.25%	74.28%
Our Road Data	30	2	8	78.94%	93.75%

We prepared a birds-eye-view window and placed each object in reference to their local coordinates, assuming that our ego vehicle is located at position $P(0,0)$. Such window can be seen in Figure 4.5. This window is divided by line pairs every 3 meters in a range of 60 meters that is resulting a grid-based map for our detected objects. In addition to that, the FOV of our vision sensor is also projected to this window. The detected vehicles from our vision sensor are placed within this area.

Table 4.4: Distance and speed comparison of camera and V2V sensors for a detected vehicle. Here *Frame No*, *Object Id*, *Distance_c*, *Distance_v*, *Speed_c*, *Speed_v*, *Xacc*, *Yacc* and *Zacc* corresponds to number of the current frame received from camera, unique id assigned by our tracker, distance estimation from camera detection, distance estimation from V2V, ego speed information from GPS receiver on IEEE 802.11p unit on ego vehicle + relative speed from camera, speed information from V2V, acceleration rate in X, Y and Z axis, respectively.

Frame No	Object Id	Distance _c	Distance _v	Speed _c	Speed _v	Xacc	Yacc	Zacc
10	2	13.50	13.68	59.27 – 0.37	55.70	0.003	0.126	-1.015
11	2	13.20	13.68	59.27 – 0.73	55.70	-0.019	0.105	-1.028
12	2	12.76	13.68	59.27 – 1.25	55.70	0.013	0.138	-1.017
13	2	12.49	13.68	59.27 – 1.58	55.70	-0.013	0.139	-1.024
14	2	12.49	13.68	59.27 – 1.58	55.70	0.020	0.119	-0.986
30	2	12.63	12.85	55.68 – 1.41	55.70	0.004	0.082	-1.047

We compared the calculated distance ($Distance_v$ in Table 4.4 and d_g in Figure 4.1) result to other pixel based distance calculation. We found that distance values are quite similar to the each other (see Table 4.4). Based on the position information we were able to match the two sensor measurements. For instance, vehicle with id 2 is detected and tracked starting from the tenth frame. We can see that our pixel based distance estimation gave the result as 13.50 meters while relative speed is estimated as -0.73 kmph. After sensor fusion, using vision sensor and V2V communication ego speed is received as 59.27 kmph and the speed of the detected vehicle 2 is received as 55.70. According to this data, we can say that our vision sensor predicted the speed of the vehicle as 58.90 kmph while V2V reported it as 55.70 kmph. The difference can be calculated roughly as %5 kmph at this frame. However, at the thirtieth frame we can measure this difference as low as %2-3 kmph. We can say that measurements of our vision sensor are reasonable and quite comparable with V2V measurements. In addition to that, we can see that the speed of this vehicle seems to have never been changed from the tenth to thirtieth frame. The time interval can be estimated as 1 second from $(30-10) / 20$ fps. We can say that our vision sensor is much more sensitive to the vehicle motion due to its high rate of sampling.

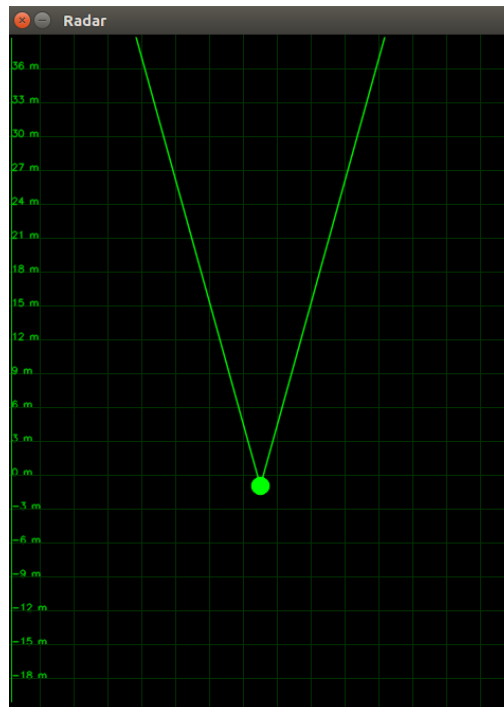


Figure 4.5: Birds-eye-view window.

The detected vehicles are also shown on each live frame that is received from the vision sensor. For instance, in Figure 4.6, one can see that three vehicles were detected and were being tracked. Our tracker algorithm gave each of them a unique id. Starting from the left side of the window, the ids of the vehicles are assigned as 3, 1, 2 and 0. These numbers are reserved and any other detected vehicle will assign next available id number, that is 4 in this case. After a vehicle is lost in vision sensor our tracker keeps the information for a certain amount of time in case the same vehicle reappears again. In our experiments, we defined this number as 3 seconds. However, since our system is also supported by the V2V communication, that vehicle is still going to be visible in our birds-eye view.

If a particular vehicle is detected by our classifier, it either draws a box around it and colors in either red or blue according to its detection score. If the score is below a certain threshold, the detected window is shown in red, otherwise it is shown in blue. This threshold value is chosen as 12 in our scenario. In Figure 4.6, we can see that cruising in vehicle in front with id 2 is detected at the fiftieth frame, however its detection score is as low as 6. It's enough for our tracker to start to track if it is detected and verified as a vehicle. One can see that vehicle with id 3 is not detected at this frame. However, it had been tracked on the previous frame, thus the tracker had started to track and keeps the tracking it even though it is not detected at the next frames.

Our tracker does not directly use the detection ROI, but instead it generates a smaller portion of that image and starts the tracking from this smaller window which is shown in blue. These windows can be seen at each tracked frame in Figure 4.6. We saw that if the detection window is directly used, the tracker tends to shift more easily as the motion in the surrounding area between the window borders and vehicle borders tend to change more.

In order to generate a better ROI window for tracking we resized the actual detection window by a certain ratio. In our case, we found that 25% of the actual window is to be considered as out of the vehicle bounds. For this reason, we generated the area of the

ROI as 75% of the detection window area. The method we used for tracking seems to have enabled us to decrease the well-known tracking shift problem.

Figure 4.7 shows the birds-eye projection of detected vehicles. We can see that the yellow car on the left and our test vehicle ahead detected and tracked successfully with object id 16 and 13, respectively. Their distance and relative speed estimation are also shown in our birds-eye view. Since our test vehicle had the IEEE 802.11p communication we were able to support its vision sensor based measurement with V2V communication. Furthermore, our system fused the two different measurements from two sensors successfully and did not generate new id for the V2V measurement, but instead it's given the same id as 13, as also can be seen in birds-eye view (white measurements from V2V, green measurements are from vision sensor).

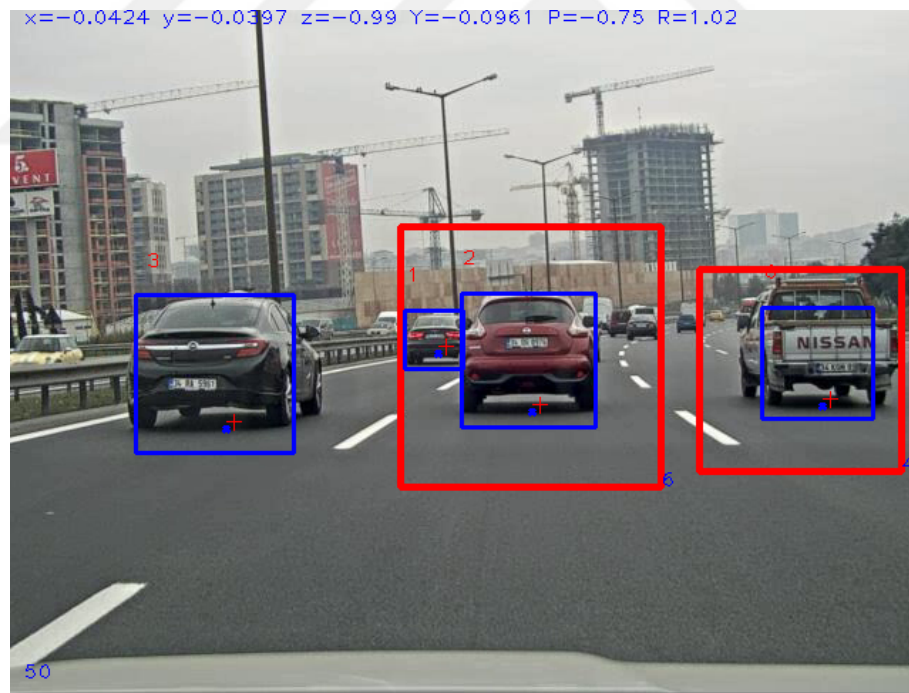


Figure 4.6: Detected vehicles.

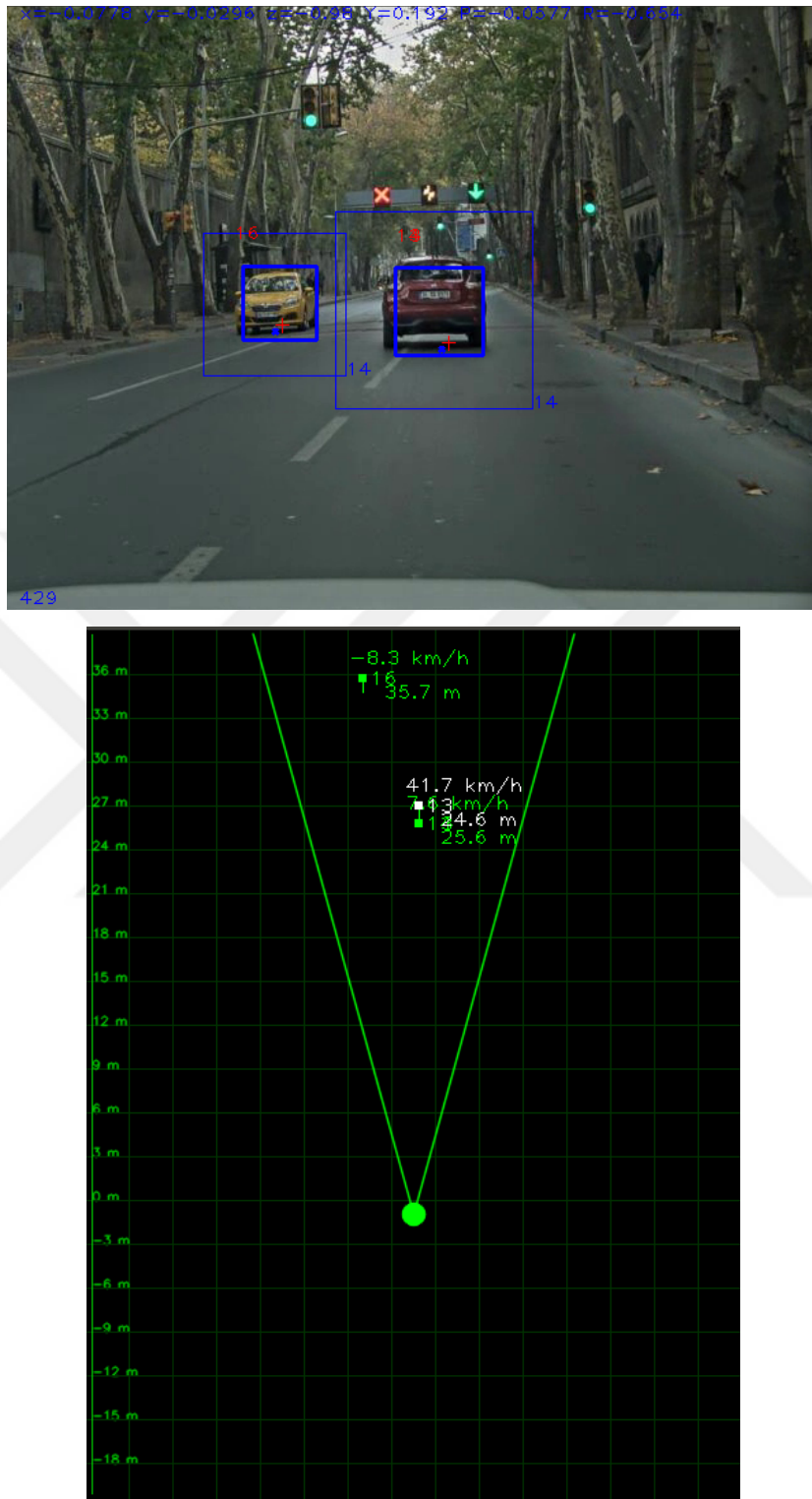


Figure 4.7: Detected vehicles and birds-eye-view projection.

In sensor fusion window in Figure 4.8, vision sensor, V2V sensor, particles and filter result shown in blue, black, violet and red, respectively. Particle filter is closer to the vision sensor than it is to the V2V sensor since its variance smaller than the V2V. Vision sensor detections can be seen in Figure 4.9. Here our test vehicle (CAR B) with id 3 cruising in front is detected from first frame until it went out of sight (field of view of the camera) at roughly 400th frame. One can also see that there are no vehicles detected for about 200 frames or 10 seconds (camera operating at 20 Hz). Here CAR B is still out of sight and cruising behind the ego CAR A. CAR B re-enters the field of view but this time with id 11, at roughly 900th frame. Figure 4.9 shows this scenario.

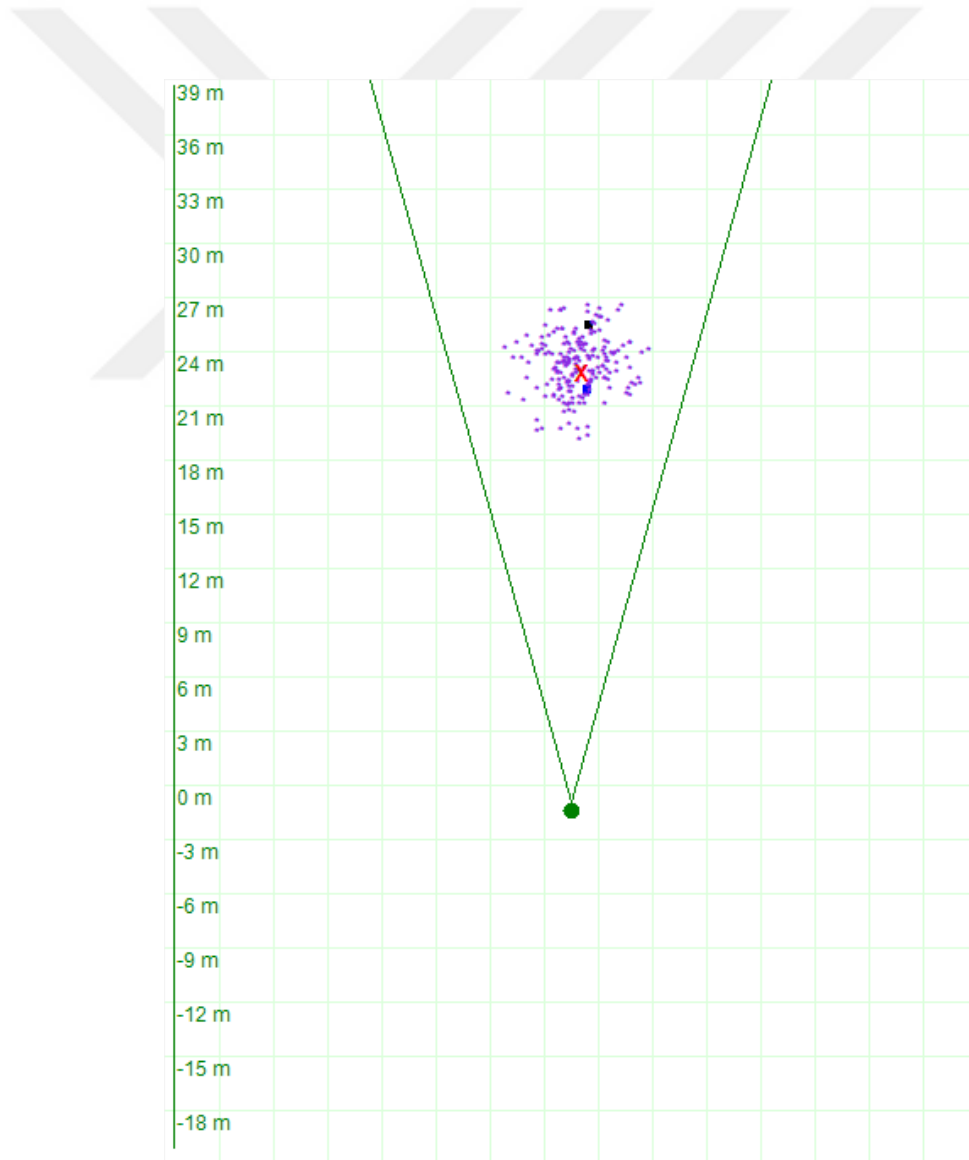


Figure 4.8: Sensor fusion view.

Even though CAR B is out of sight of the vision sensor in Figure 4.9 (b), one can see that our system still detects and tracks it with the IEEE V2V sensor as shown in Figure 4.10. However, particle filter results are tend to close to the IEEE V2V sensor measurements where we do not have the vision sensor detection. These scenarios are also shown in charts in Figure 4.11. It is clear that particle filter outputs promising results where IEEE V2V sensor measurements shifts dramatically.

Our full dataset is summarized in Table 4.5. We collected 12 different dataset in Istanbul covering more than 45 km of road and 65 minutes of data. Used road data in our experiments is from the 8th row in Table 4.5. Path of this data is also shown on Yandex Maps in Figure 4.12.

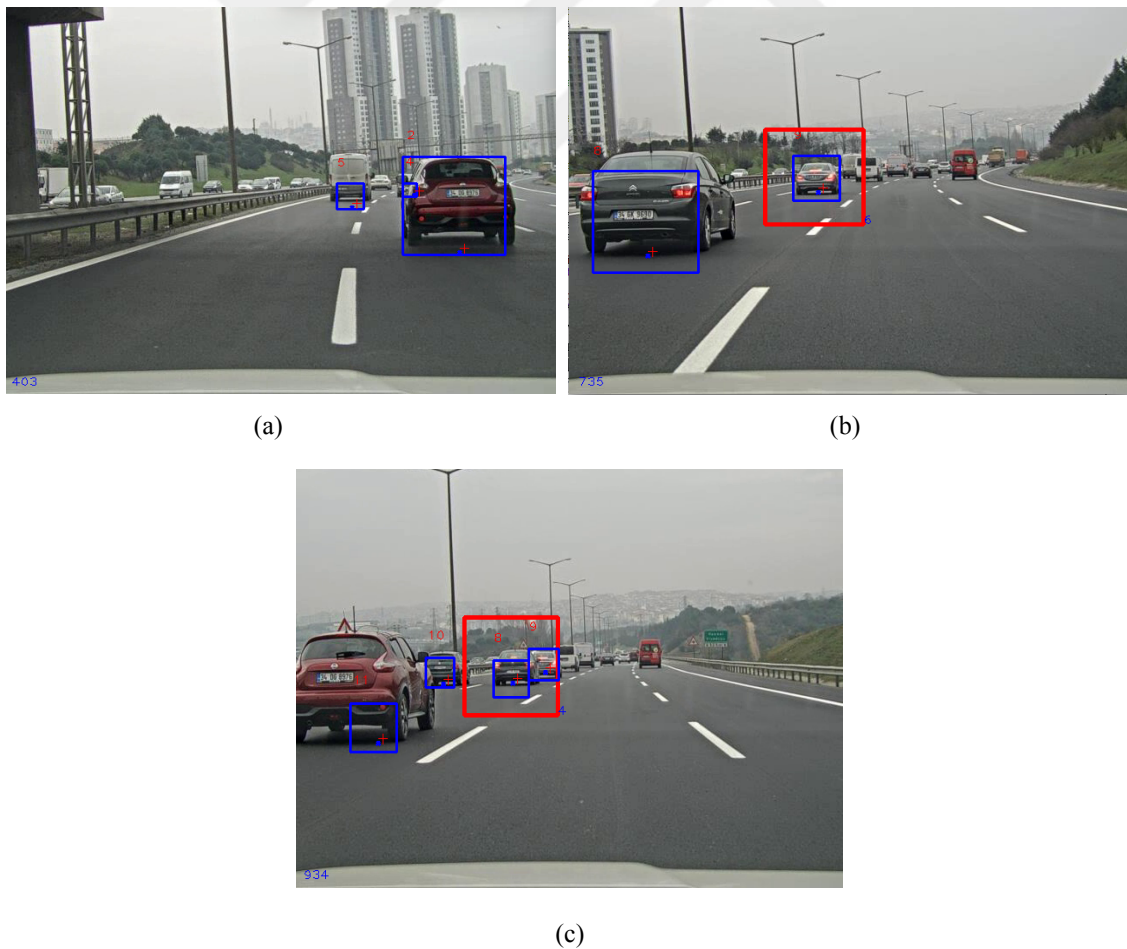


Figure 4.9: Vision sensor detections. Numbers refer to the Id of the vehicles.

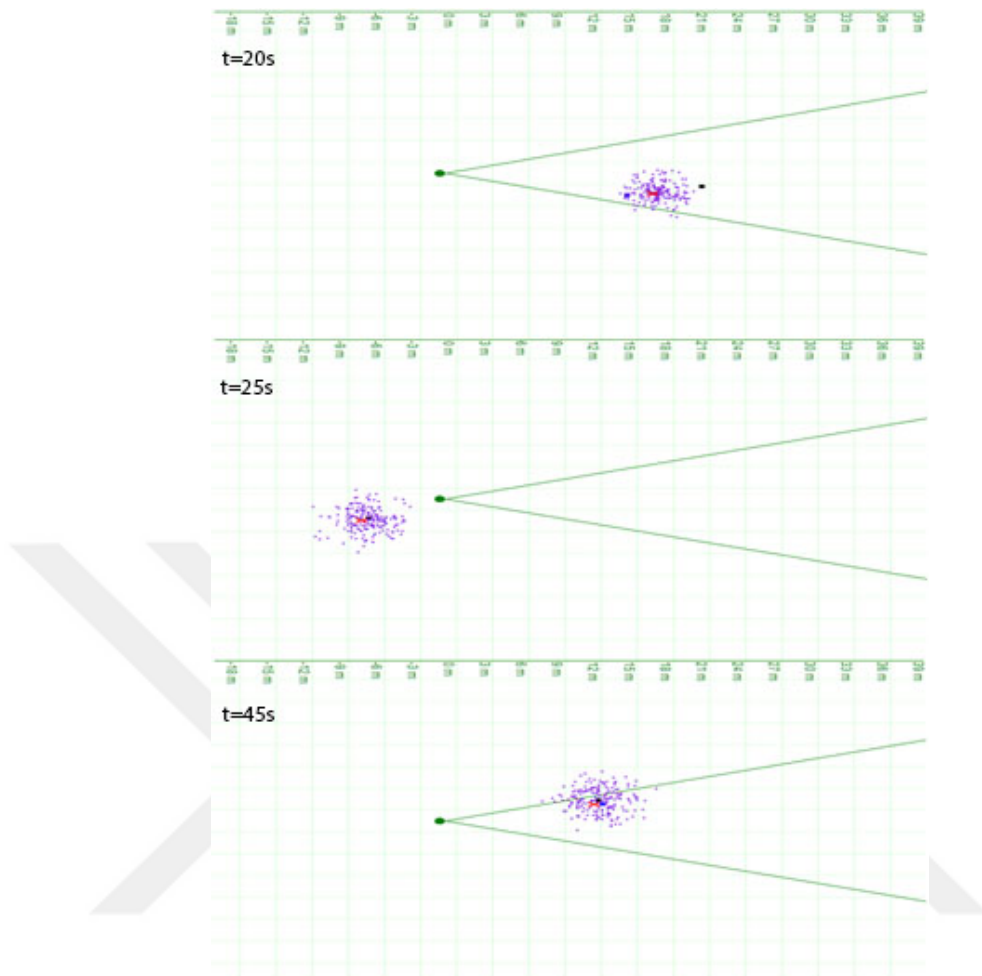


Figure 4.10: Out-of-sight scenario. $t=20s$: CAR A is about to overtake CAR B. $t=25s$: CAR B is behind the CAR A. $t=45s$: CAR B re-enters the field of view of the camera.

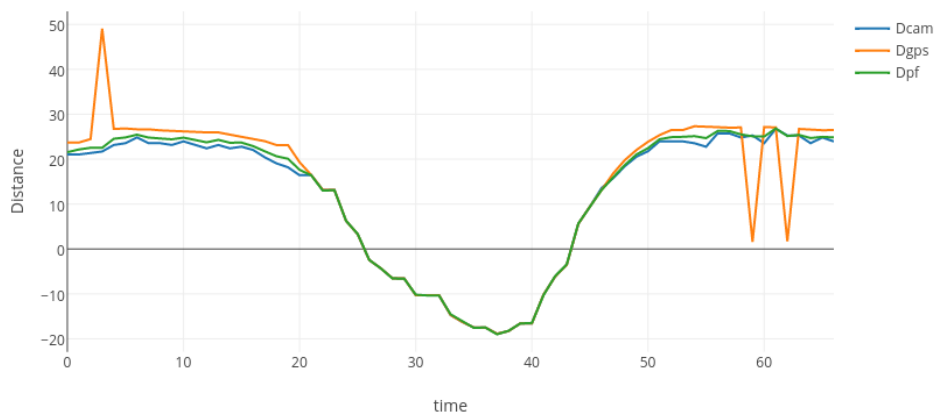


Figure 4.11: Particle filter result. D_{cam} , D_{gps} and D_{pf} refer to vision sensor measurement, GPS measurement from V2V sensor and particle filter result, respectively.

Table 4.5: Summary of our collected road data.

Odometer (km)	Lanes	Traffic	Road Type	Avg. Speed (km/h)
3.5	1-2	Free	Urban	40
0.75	1-2	Free	Urban	20
1.6	1-2	Free	Urban	45
1.8	1-2	Congested-Free	Urban	20
2.7	1-3	Congested-Free	Urban	20
5.1	3	Congested-Free	Highway	60
6.9	2-4	Free	Highway	80
4.2	2-4	Free	Highway	60
12.0	2-4	Free	Highway	75
1.6	1-2	Congested-Free	Urban	30
0.65	1	Congested	Urban	5
6.4	1-2	Congested	Urban	30

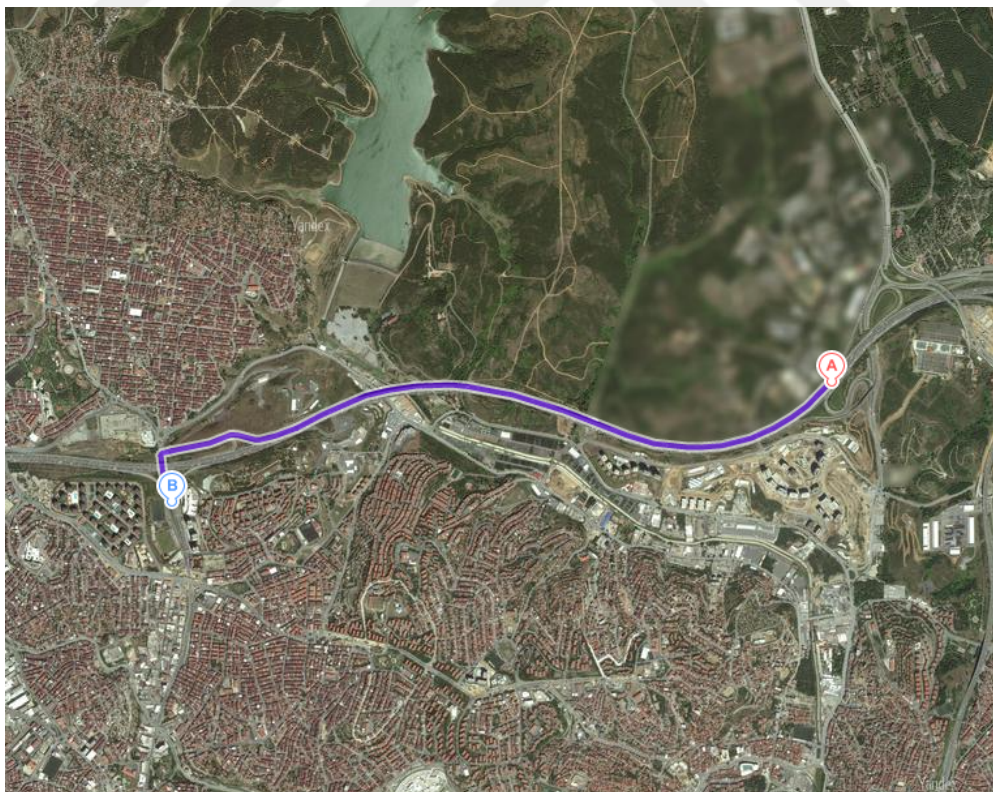


Figure 4.12: Path of the road data used in experiment (Table 4.5 8th row)

For the real time capability of the system, we also test our system in the GPU accelerated NVIDIA Tegra Jetson TK1 development board. This board is the simpler developer version of the industrial GPU accelerated boards¹. NVIDIA Tegra Jetson TK1 has the NVIDIA Kepler “GK20a” GPU with 192 SM3. 2 CUDA cores (up to 326 GFLOPS), NVIDIA “4-Plus-1” 2.32 GHz ARM quad-core Cortex-A15 CPU and 2GB DDR3L 933 MHz RAM. Figure 4.13 shows the diagram of such board.

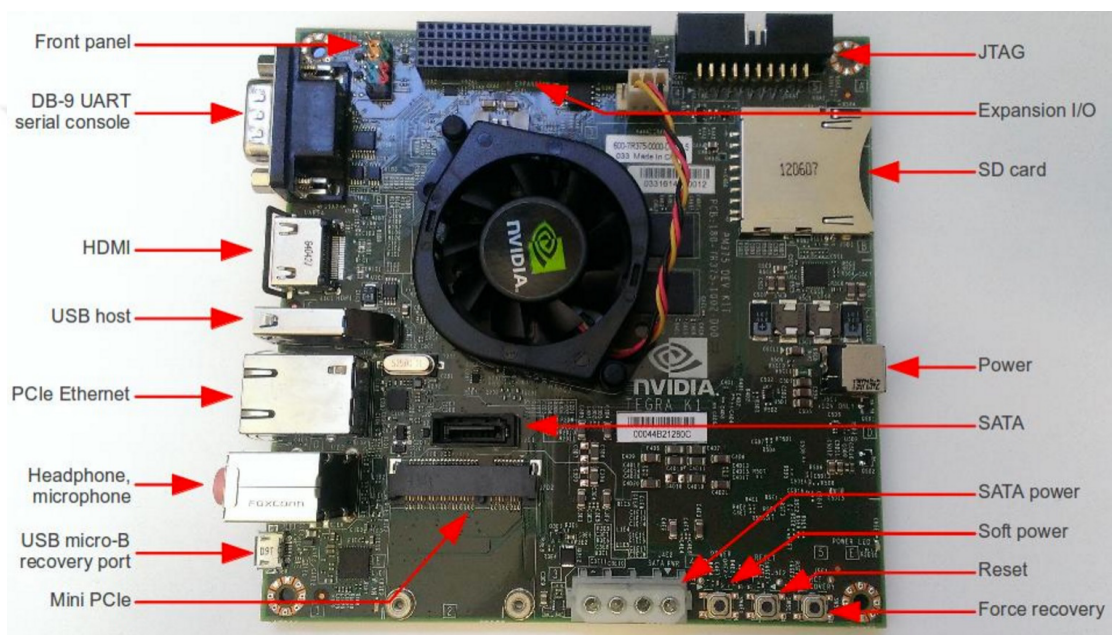


Figure 4.13: Hardware features of NVIDIA Jetson TK1 board².

We tested Canny edge detection and the Hough Lines Transform algorithm which are used in our system on both CPU and GPU and compared their results. We saw that if there is no iteration and the image to be processed is not big enough, uploading it to GPU and processing it is more expensive than the CPU implementation. Figure 4.14-4.16 shows this result. We did not take the necessary time of initial upload of the image to the GPU into account which took about additional 2 seconds, as also can be seen in Figure 4.14.

¹ <http://www.nvidia.com/object/jetson-tx1-module.html>

² http://elinux.org/Jetson_TK1

```

ubuntu@tegra-ubuntu: ~
ubuntu@tegra-ubuntu:~$ ./build-GPUTESTS-De
loads/highway1.jpg true 320 240
Picture size:3888x2592
Picture resized to:320x240
CPU Time : 55.1178 ms
CPU Found : 21
CPU Time : 47.9232 ms
CPU Found : 22
CPU Time : 43.6473 ms
CPU Found : 17
CPU Time : 40.4138 ms
CPU Found : 13
CPU Time : 37.6394 ms
CPU Found : 11
CPU Time : 38.4025 ms
CPU Found : 11
CPU Time : 35.3286 ms
CPU Found : 8
CPU Time : 34.197 ms
CPU Found : 7
CPU Time : 34.4549 ms
CPU Found : 7
CPU Time : 32.4288 ms
CPU Found : 2
Average CPU Time:39.9553 ms
Average CPU Found:11
GPU upload time: 2.11336
GPU Time : 89.7734 ms
GPU Found : 78
GPU Time : 19.4909 ms
GPU Found : 78
GPU Time : 26.7998 ms
GPU Found : 78
GPU Time : 25.688 ms
GPU Found : 78
GPU Time : 20.3935 ms
GPU Found : 78
GPU Time : 29.8782 ms
GPU Found : 78
GPU Time : 20.4866 ms
GPU Found : 78
GPU Time : 23.1699 ms
GPU Found : 78
GPU Time : 18.6162 ms
GPU Found : 78
GPU Time : 26.7466 ms
GPU Found : 78
Average GPU Time:30.1043 ms
Average GPU Found:78

ubuntu@tegra-ubuntu: ~
ubuntu@tegra-ubuntu:~$ ./build-GPUTESTS-De
loads/highway1.jpg true
Picture size:3888x2592
CPU Time : 1636.5 ms
CPU Found : 88
CPU Time : 1494.13 ms
CPU Found : 102
CPU Time : 1494.22 ms
CPU Found : 84
CPU Time : 1496.01 ms
CPU Found : 95
CPU Time : 1525.74 ms
CPU Found : 83
CPU Time : 1493.77 ms
CPU Found : 68
CPU Time : 1509.36 ms
CPU Found : 72
CPU Time : 1506.35 ms
CPU Found : 62
CPU Time : 1502.69 ms
CPU Found : 49
CPU Time : 1503.31 ms
CPU Found : 52
Average CPU Time:1516.21 ms
Average CPU Found:75
GPU upload time: 2.02483
GPU Time : 764.594 ms
GPU Found : 2014
GPU Time : 739.869 ms
GPU Found : 2014
GPU Time : 721.546 ms
GPU Found : 2014
GPU Time : 727.224 ms
GPU Found : 2014
GPU Time : 723.938 ms
GPU Found : 2014
GPU Time : 747.099 ms
GPU Found : 2014
GPU Time : 734.057 ms
GPU Found : 2014
GPU Time : 751.353 ms
GPU Found : 2014
GPU Time : 729.181 ms
GPU Found : 2014
GPU Time : 728.883 ms
GPU Found : 2014
Average GPU Time:736.774 ms
Average GPU Found:2014

```

Figure 4.14: Canny edge detection and Hough Transform performance on resized (320x240) image (left) vs the original (3888x2592) size (right).

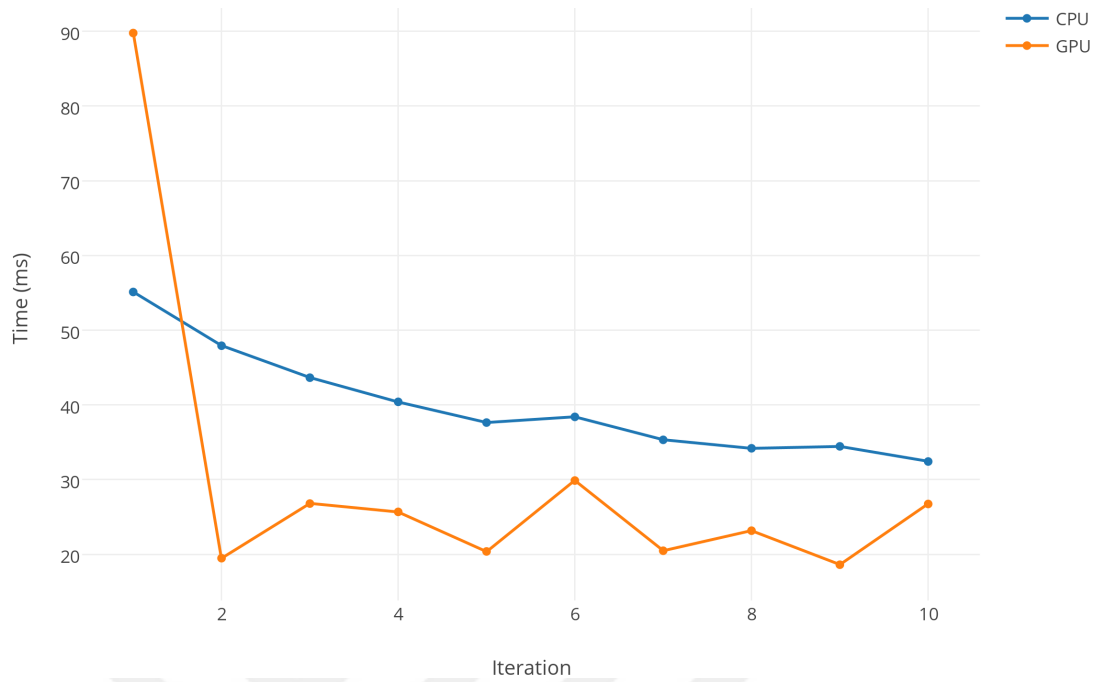


Figure 4.15: Canny edge detection and Hough Transform performance on resized image (320x240).

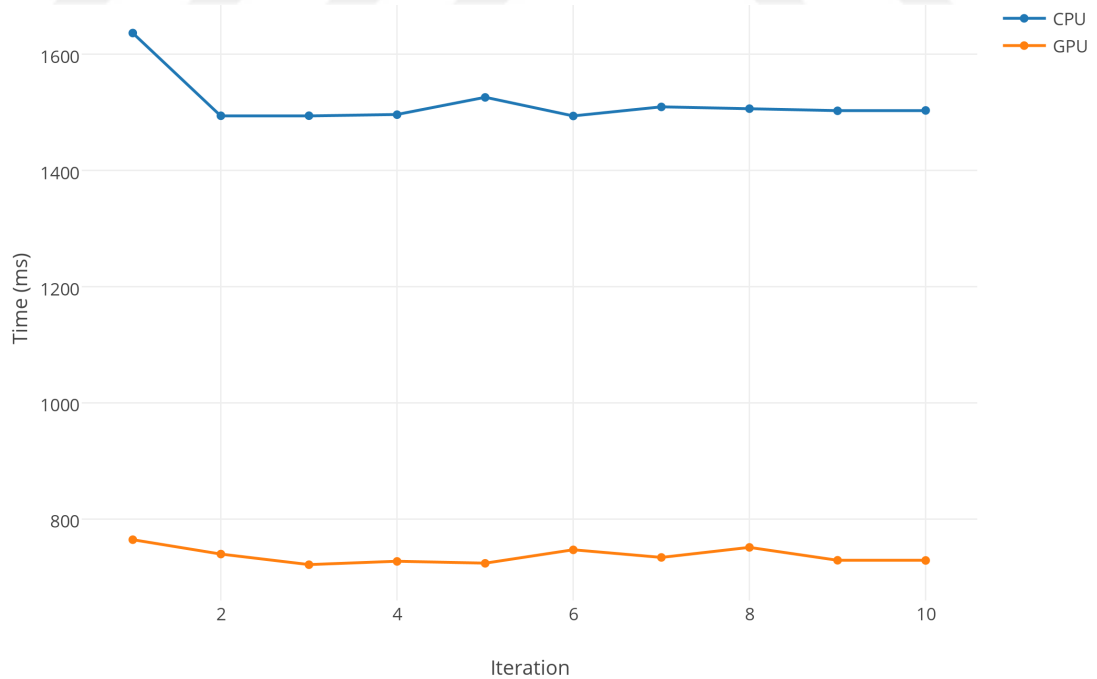


Figure 4.16: Canny edge detection and Hough Transform performance on original image (3888x2592)

When we have the iteration, it seems that the overall performance of GPU outperforms the CPU with these algorithms. It can be seen that when the image size is about 10 MP (Figure 4.14 right) the GPU implementation has speeded up the whole process by 2x. One can also see that the number of the detected lines on the same image is approximately 27x more than the CPU implementation. However, when the image is resized to 320x240 pixels, we can see that the speed difference and the number of lines detected are dropped to 1.3x and 7x, respectively. In addition to speed up, these results show that we also obtained much more lines detected in comparison to CPU. This may be useful in future tasks. The outputs of this comparison can be seen on Figure 4.17.



Figure 4.17: Output of CPU vs GPU line detection test on the highway image³

³ https://en.wikipedia.org/wiki/Ontario_Highway_407

5. RESULTS & CONCLUSIONS

As a conclusion, we focused on monocular based driver assistance system, which is a core technology for autonomous vehicles. We enhanced our system with IEEE 802.11p V2V communication system. We trained our classifier with 550 positive and 500 negative samples from UIUC image database. We tested the detection performance of our system with our own dataset and compared with KITTI vision benchmark dataset. We saw that we received 93.75% and 74.28% precision rate, 78.94% and 81.25% recall rate with our dataset and KITTI dataset, respectively.

A modern tracker system has been adapted for tracking. A unique id was assigned for each tracked vehicle. This method has enabled us to track the vehicles even though our detector failed at some cases. We implemented the pixel based distance estimation. Distance estimation from our vision was a key estimation for our fusion system. Here we assumed that the road is flat or ego and the detected vehicle have the same pitch. Pixel based distance estimation and Median Flow tracker together, gave us the ability for relative speed estimation.

Then we verified the detection of our test vehicle with V2V communication to be fused with the vision sensor. We matched measurements from different sensors based on the Euclidean distances for a particular detected vehicle. Since both sensors have errors, we used particle filter algorithm. For our tests, we chose 3 and 4 meters variance for vision sensor and GPS measurement from V2V communication, respectively.

In order to evaluate our system's performance in real road conditions, we collected real road data from both urban roads and highways of Istanbul. Our dataset consist of 12 different segments, including over 47 kilometers and 60 minutes of drive data. We also

classified our road data based on lane count and traffic status as 1-4 lanes and Free-Congested-Free, Congested, respectively.

After we test our system with our dataset, our system enabled us to sense the vehicles even when they are out of sight or occluded by other objects. Specifically, we saw that when the detected vehicle is out of the field of view of the vision sensor, V2V communication still enabled us to detect and track it successfully. Particle filter algorithm gave promising results even GPS measurements shifted dramatically and even vision sensor information was not present due to out-of-sight vehicle, at our scenario. This way our system had the capability of sensing the vehicles even when they are occluded or even not present in vision sensor measurements.

Lastly, due to real time requirement of this kind of application we run some of the algorithms we used (Canny edge detection and Hough transform) in GPU accelerated NVIDIA Jetson TK1 board. We saw that we could speed up our system, especially at higher resolutions.

For the future work, we intend to increase our detection rate by applying a better hypothesis extraction and verification. In addition, we used two vehicles for our real road tests, for better understanding of the outputs of our system, we need more real traffic data from more vehicles. Moreover, our distance measurements had no truth reference. For this reason, we also plan to equip or ego test vehicle with LIDAR sensor and get the ground truth reference for our system outputs.

REFERENCES

- Agarwal S, Awan A. Roth D. UIUC Image Database for Car Detection. URL: <https://cogcomp.cs.illinois.edu/Data/Car/>
- Alessandretti G, Broggi A, Cerri P. (2007). Vehicle and Guard Rail Detection Using Radar and Vision Data Fusion, IEEE Transactions on Intelligent Transportation Systems: Vol.8, No.1: 95-105
- Bertozzi M, Broggi A. (1998). GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection, IEEE Transactions on Image Processing: vol.7, no.1: 1998, pp.62-81.
- Betke M, Haritaoglu E, Davis L. S. (2000). Real-time multiple vehicle detection and tracking from a moving vehicle, Machine Vision and Applications, New York: Springer-Verlag, pp.69-83.
- Canny J, (1986). A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence: 8(6), pp.679-698.
- Caraffi C, Vojir C, Trefny J, Sochman J, Matas J. (2012). A System for Realtime Detection and Tracking of Vehicles from a Single Car-mounted Camera, 15th Int. Conf. on Intelligent Transportation Systems, Hilton Anchorage, Alaska, USA, pp.2019-2034.
- Chen Y, Wu B, Huang H, Fan C. (2011). A Real-Time Vision System for Nighttime Vehicle Detection and Traffic Surveillance, IEEE Transactions on Industrial Electronics: vol.58, no.5: 2011, pp.2030-2044.

- Choi J, Lee J, Kim D, Soprani G, Cerri P. (2012). Environment-Detection and Mapping Algorithm for Autonomous Driving in Rural or Off-Road Environment, IEEE Transactions on Intelligent Transportation Systems: vol.13, no.2: 2012, pp.974-982.
- Çayır B, Acarman T. (2009). Low Cost Driver Monitoring and Warning System Development, Intelligent Vehicles Symposium, IEEE, Xi'an, China, pp.94-98.
- ETSI EN 302 571 (V1.1.1) (2008). Intelligent Transport Systems (ITS); Radiocommunications equipment operating in the 5 855 MHz to 5 925 MHz frequency band; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive. URL: http://www.etsi.org/deliver/etsi_en/302500_302599/302571/01.01.01_60/en_302571v010101p.pdf
- Geiger A, Lenz P, Stiller C, Urtasun R. (2013). Vision meets robotics: The KITTI dataset, The International Journal of Robotics Research: vol.32, no.11: 2013, pp.1231-1237.
- Huo C, Yu Y, Sun T. (2012). Lane Departure Warning System based on Dynamic Vanishing Point Adjustment, The 1st IEEE Global Conference on Consumer Electronics, Makuhari Messe, Tokyo, Japan, pp.25-28.
- Jazayeri A, Cai H, Zheng J, Tuceryan M. (2011). Vehicle Detection and Tracking in Car Video Based on Motion Model, IEEE Transactions on Intelligent Transportation Systems: vol.12, no.2: 2011, pp.583-595.
- Kalal Z, Mikolajczyk K, Matas J. (2010). Forward-backward error: Automatic detection of tracking failures, IEEE 20th International Conference on Pattern Recognition, Istanbul, Turkey, pp.2756-2759.
- Kim H, Song B. (2013). Vehicle Recognition Based on Radar and Vision Sensor Fusion for Automatic Emergency Braking, 13th International Conference on Control, Automation and Systems, Kimdaejung Convention Center, Gwangju, Korea, pp.1342-1346.
- Korkmaz G, Ekici E, Özgüner F. (2006). An Efficient Ad-Hoc Multi-Hop Broadcast Protocol for Inter-Vehicle Communication Systems, IEEE International Conference on Communications, Istanbul Congress & Exhibition Center, Istanbul, Turkey, pp.423-428.

- Lienhart R, Kuranow A, Pisarevsky V. (2003). Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection, Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, pp.297-304.
- Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, 60, 2, pp.91-110.
- Matas J, Galambos C, Kittler J.V. (2000). Robust Detection of Lines Using the Progressive Probabilistic Hough Transform, CVIU 78 1, pp.119-137.
- Rezaei M, Terauchi M, Klette R. (2015). Robust Vehicle Detection and Distance Estimation Under Challenging Lighting Conditions, IEEE Transactions on Intelligent Transportation Systems: vol.16, no.5:2015, pp.2723-2743.
- Shiraki Y, Ohyama T, Nakabayashi S, Tokuda K. (2001). Development of an Inter-Vehicle Communications System, OKI Technical Review 187: vol.68,: 2001, pp.11-13.
- Sun Z, Bebis G, Miller R. (2006). Monocular Precrash Vehicle Detection: Features and Classifiers, IEEE Transactions on Image Processing: vol.15, no.7: 2006, pp.2019-2034.
- Tokuda K, Masami A, Fujii H. (2000). DOLPHIN for Inter-Vehicle Communications System, IEEE Intelligent Vehicles Symposium, Dearborn, Miami, USA, pp.504-509.
- Tsugawa S, Kato S, Tokuda K, Matsui T, Fujii H. (2001). A Cooperative Driving System with Automated Vehicles and Inter-Vehicle Communications in Demo 2000, IEEE Intelligent Transportation Systems Conference, Oakland, California, USA, pp.918-923.
- Viola P, Jones M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii, USA, pp.I-511-I-518.
- Yıldız R, Acarman T. (2012). Image Feature Based Video Object Description and Tracking, IEEE International Conference on Vehicular Electronics and Safety, İstanbul, TURKEY, pp.405-410.

BIOGRAPHICAL SKETCH

Mustafa Tekeli was born on March 15, 1987 in Ankara. He received his high school education in Istanbul Tarabya Yaşar Dedeman High School. Furthermore, he received his Bachelor of Science in Computer Engineering from Beykent University in 2012.

