A MULTIMODAL SENSOR ANALYSIS FRAMEWORK FOR VEHICULAR MOBILE APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

FATİH ORHAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2013

A MULTIMODAL SENSOR ANALYSIS FRAMEWORK FOR
VEHICULAR MOBILE APPLICATIONS

Submitted by **FATİH ORHAN** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems, Middle East Technical University** by,

Prof. Dr. Nazife BAYKAL
Director, **Informatics Institute** _____

Prof. Dr. Yasemin YARDIMCI ÇETİN
Head of Department, **Information Systems** _____

Assist. Prof. Dr., P. Erhan EREN
Supervisor, **Informatics Institute, METU** _____

Examining Committee Members:

Assoc. Prof. Dr., Altan KOÇYİĞİT
Information Systems, METU _____

Assist. Prof. Dr., P. Erhan EREN
Information Systems, METU _____

Dr., Nail ÇADALLI
KAREL A.Ş. _____

Assist. Prof. Dr., Banu GÜNEL
Information Systems, METU _____

Assist. Prof. Dr., Alptekin TEMİZEL
Work Based Learning, METU _____

Date: 04.09.2013

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Fatih ORHAN

Signature : _____

ABSTRACT

A MULTIMODAL SENSOR ANALYSIS FRAMEWORK FOR
VEHICULAR MOBILE APPLICATIONS

Orhan, Fatih
M. Sc., Department of Information Systems
Supervisor: Assist. Prof. Dr., P. Erhan Eren

September 2013, 72 pages

The sensing, computing and communicating capabilities of smart phones bring new possibilities for creating remarkable applications increasing the quality, safety, comfort, economy and other capabilities of cars. However, many challenges exist regarding the development of multimodal sensor analysis applications, such as proper collection of sensor values, integration of diverse libraries and tools for sharing the results. This study focuses on these challenges and aims to construct a framework that enables easy, fast and flexible implementations of smart vehicular applications.

The goal is to provide capabilities for real-time sensing, signal and multimedia processing, and sharing of information. Hence, the proposed framework provides an abstraction for easy access to sensor readings, and also diverse signal and image processing libraries are integrated and mechanisms are developed and provided with the framework for information sharing. A sample mobile application is also developed in the scope of this study, in order to demonstrate the capabilities and the feasibility of the developed framework. This application is a multimodal sensor analyzer which intends to detect obstacles on the road or critical road surface anomalies (i.e. pothole, speed bump) by measuring and analyzing different sensors of smart phones (GPS, accelerometer, magnetometer and camera) in real-time. First, the detection is performed on one modality by analyzing the motion of the vehicle using the accelerometer sensor and then a second analysis is performed using the camera images to automatically extract the video section and the image of the corresponding road segment containing the defect. Upon such critical hazard detection, the application instantly informs nearby users about the incident with detailed information and image of the scene.

The developed mobile application is deployed on multiple devices as part of a test scenario in a chosen location and the outcomes of the test scenario are measured and evaluated as part of the validation.

Keywords: multimodal sensor analysis, mobile GIS applications, pervasive computing, vehicular mobile application, mobile multimedia.

ÖZ

ARAÇ İÇİ MOBİL UYGULAMAR İÇİN ÇOK-KİPLİ ALGILAYICI ANALİZ ALTYAPISI

Orhan, Fatih

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Yrd. Doç. Dr., P. Erhan Eren

Eylül 2013, 72 sayfa

Akıllı telefonların gelişmiş algılama, hesaplama ve iletişim yetenekleri araçlarda kalite, güvenlik, konfor, ekonomi ve diğer yetenekleri arttıran faydalı uygulamalar geliştirme imkânı doğurmaktadır. Ancak çoklu sensör analizi uygulamaları geliştirmek için algılayıcı değerlerini doğru okuma, çeşitli kütüphaneleri bütünleştirme ve elde edilen sonuçları paylaşma gibi birçok zorluk da bulunmaktadır. Bu çalışma bahsi geçen zorluklara odaklanmakta ve araç içi akıllı uygulamaların kolay, hızlı ve esnek gerçekleştirimini sağlayacak bir altyapı oluşturmayı hedeflemektedir.

Amaç; gerçek zamanlı algılama, sinyal ve çoklu-ortam işleme ve bilgi paylaşma yeteneklerini sağlamaktır. Bu sebeple önerilen altyapı sensör değerlerine hızlı erişim için bir soyutlama sağlamakta, ayrıca sinyal ve görüntü işleme için çeşitli bütünleştirilmiş kütüphaneler sunmakta ve bilgi paylaşımı için oluşturulmuş mekanizmalar altyapı ile birlikte sunulmaktadır.Bu çalışma kapsamında, geliştirilen altyapının yeteneklerini ve uygulanabilirliğini göstermek amacıyla örnek bir mobil uygulama da geliştirilmiştir. Bu uygulama akıllı telefonların farklı algılayıcılarını (örn. GPS, ivmeölçer, manyetometre, kamera) gerçek zamanlı olarak ölçüp analiz eden ve yoldaki engelleri, yol yüzeyinin düzensizliklerini (örn. çukur, kasis) tespit eden çok-sensörlü bir analizördür. Öncelikle düzensizliğin tespiti bir kipte mobil cihazın ivmeölçer algılayıcısı kullanılarak aracın hareketi analiz edilerek gerçekleştirilmekte, ikinci bir analiz ile de kameranın kayıtları kullanılarak yolun bozuk olan kısmına ait video kesiti ve fotoğraf otomatik olarak çıkarılmaktadır. Bir tehlike tespit edildiğinde sistem olası kazaları veya istenmeyen olayları engellemeye yönelik olarak coğrafi konumu yakın olan diğer sürücüleri olaya ait detaylı bilgi ve fotoğraf ile uyarabilmektedir.

Geliştirilen mobil uygulama belirlenen bir alandaki bir test senaryosu kapsamında birçok cihaz üzerine kurulmuş ve doğrulama kapsamında test senaryonun sonuçları ölçülmüş ve değerlendirilmiştir.

Anahtar kelimeler: çok-sensörlü analiz, mobil CBS uygulamaları, yaygın bilişim, araç içi mobil uygulama, mobil çoklu-ortam.

To My Wife

For her endless support,

And her love…

ACKNOWLEDGEMENTS

First of all, I would like to thank my adviser Assistant Professor P. Erhan Eren for his support, the time he spent to advice and guide me how to create valuable functional ideas and transform them in the work of this thesis. He always aligned me to the importance of exercising intellectual and technical curiosity as the only means to achieve academic impact. By challenging ideas and assumptions, he fostered a sense of criticism and the need to think outside of the box, a must-have characteristic to be a researcher.

I owe immense gratitude to my wife, for her patience to put up with my busy work and education life. Her presence and strength always give me the drive to accomplish such challenges.

I must express my tremendous appreciation to my dear parents and family members, who always supported me during my whole education. Thank you for your love and support.

Finally, my special thank is for my twins, for providing motivation and giving me the pleasure to take care of and play with them. They always reminded me what is really important.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ALS	Ambient Light Sensor
CPU	Central Processing Unit
DAS	Driver Assistance Systems
DVR	Digital Video Recorder
EDR	Event Data Recorder
GCM	Google Cloud Messaging for Android
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IRI	International Roughness Index
ITS	Intelligent Transportation System
JNI	Java Native Interface
METU	Middle East Technical University
NDK	Native Development Kit
SDK	Software Development Kit
SMS	Short Message Service

CHAPTER 1

INTRODUCTION

The convergence of mobile technologies enabled the smart phones to provide various usages due to their broad and diverse sensing capabilities. The location, motion and camera sensors (as well as other sensors) are commonly utilized to realize functional pervasive mobile applications. One of such emerging application areas is vehicular applications, in which smart systems assist drivers to increase their security, facilitate their journey in traffic and/or provide entertainment facilities.

The pervasive mobile computing is thriving; however there are remarkable challenges to produce smart vehicular applications, specifically due its dynamic environment. In order to implement a smart application, one has to overcome the challenges including but not limited to: proper collection of sensor values, integration of diverse libraries for multimodal sensor analysis and tools for sharing the results.

This study presents a mobile sensor framework, with a significant focus on vehicular applications that aims to:

- Enable easy and flexible usage of mobile device sensors, including camera,
- Introduce tools and methods for implementing analyzers of these sensor values, principally for signal and multimedia processing, and
- Provide means of sharing results deduced from the analysis.

The study includes also an implementation of a sample application using the developed framework. The application may be defined as a multimodal analyzer system in order to detect hazardous incidents in traffic and inform other drivers about the event by providing details of the incident like the location, the type of the incident and possibly a snapshot photo. The system comprises of a regular smart phone having GPS, accelerometer and camera sensors mounted on the front part in the car. The multimodal analysis is performed on device and in real time. The drivers nearby, using the same system, are informed about the hazardous event using a push mechanism via a central server. This system tries to prevent new incidents to occur and also helps drivers to have a safe trip with their vehicles.

The last output of this study is a set of sensor values from GPS, accelerometer and magnetometer sensors as well as video records. The sensor outputs are collected while driving in a selected test site by various users using different mobile devices and different vehicles.

1.1 Motivation

Technologic developments improve the quality of our lives by introducing new tools and methods day-by-day. Health, security, communication and transportation are the major topics that governments, academia and private sector focus to obtain innovative, practical and profitable products and services.

Computing and sensing technologies are two of such technologies and they are evolved much in the last decades. With the drop of production prices, our physical environment is flooded by different types and capabilities of sensing and computing devices. Homes, offices, cars as well as personal belongings are invaded with integrated circuits and/or sensors that are equipped to facilitate our daily lives, by sometimes raising warnings to avoid accidents, by sometimes conducting small tasks on behalf of us or by just performing any other assistive function that helps us.

The main ingredient of sensors is the contextual data. They measure and obtain small, bit and piece of data about the environment they perceive. This data is usually valid and functional for only a limited time of period, but if used in this time period, it may create a high value to the user. The reason why sensors are being integrated into our lives lies in this fact: they provide contextual time-bounded information and information "is subject to just-in-time requirements, just like physical inventory. Left on its own, its value may depreciate over time" [21].

Intelligent Transportation Systems (ITS) is a promising domain for sensing and computing technologies and many companies make important investments[1,2] on Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) communications. The idea of tomorrow's cars aims to increase the safety of driver and passengers as well as to lower fuel consumption and gas emissions while providing a comfortable and entertaining driving experience.

Considering the vehicular traffic context, information about traffic congestion just ahead of a driver's path is probably highly valuable. An in-car system that warns the driver about upcoming possible threats or incidents on the road would also be worthwhile for most drivers. A video and data recording system as a personal traffic surveillance application is both functional for post incident analysis and also for personal recordings of places visited helping to create a driving diary. If, for example, information about traffic incidents would be conveyed "just-in-time" to nearby vehicles, then many accidents would possibly be prevented or at least damages would be lessened.

[1] Cisco and NXP invest in Cohda, will work together to enable connected car http://www.engadget.com/2013/01/04/cisco-and-nxp-invests-in-cohda/

[2] The Volvo's SARTRE Project http://www.sartre-project.eu/en/Sidor/default.aspx

In 2011, 1,228,928 traffic accidents occurred in Turkey [20], and more than %10 of these accidents were fatal. The fatality rate for 100,000 vehicles is 23.8 people and injury rate is 1.479 people [20]. The case is not much different in developed countries. In US, over 30,000 people are killed in accidents each year [22] and the rate for 100,000 vehicles is an average of 14.5 people for the year 2008 [23]. These crashes total cost was more than $230.6 billion in 2000 [25], more than a decade ago. The report of U.S Census Bureau [23] states also that 16% of fatal crashes and 20% of injuries occurred due to driver distraction[3].

The distraction rate is important, because the sensing technology helps the driver to be aware of his/her environment, by providing contextual information. The sensors would increase the consciousness of the driver about upcoming threat for instance, supposed that a threat is detected by the sensor. As in this example, sensing and computing technologies currently enable the development of systems for secure integrated traffic systems. Intelligent Traffic Systems (ITS), smart vehicles, Driver Assistance Systems (DAS), in-car entertainments systems and many other concepts are developed in order to increase quality, safety, comfort, economy or other capabilities of cars. The future of car development is envisioned as “clean, accident-free and autonomously driving car” [24].

On the other hand, smart phone device adoption is increasing rapidly and Flurry [26] reports that more than 640 million iOS and Android devices were in use in July 2012 all over the world. Most of these devices contain highly capable sensing components. The leading operating systems in mobile computing are currently iOS (owned by Apple Inc.) and Android (owned by Google Inc.). They both provide open and flexible Software Development Kits (SDK), which permit the development of native mobile applications.

Today’s typical smart phone includes sensors including: GPS, accelerometer, magnetometer, gyroscope, light, microphone, camera (front and rear), proximity, Bluetooth, WiFi, GSM, touchscreen and much more. The access to device sensors is a de-facto for the mentioned platforms, and the developers are encouraged to develop context-based smart applications using the sensors. The delivery of the developed applications is also straightforward. The developers only deploy their applications to application stores, and all users may download it instantly without any hurdle.

Although the computing and sensing technology flourished, and the required infrastructures are setup and ready, there are various challenges to develop smart contextual applications using sensors. Due to its dynamic environment and broad requirements, vehicular mobile application development is much more complicated. Briefly, these complications may be listed as follows:

- Proper collection of sensor values: the sensors are various and each of them may provide different interfaces to access the generated values. Real-time processing of the sensor values are also important and generally vehicular applications require prompt processing of sensor values. On the other hand,

[3] “‘Distraction’ is defined as a specific type of inattention that occurs when drivers divert their attention from the driving task to focus on some other activity instead. It is worth noting that ‘distraction’ is a subset of ‘inattention’ (which also includes fatigue, physical conditions of the driver, and emotional conditions of the driver).” [23].

multimodal sensor analysis requires a fusion of the sensor values, for which the prerequisite is usually the synchronization of different sensor values.

- Integration of diverse libraries for multimodal sensor analysis: Signal processing or image processing techniques should generally be applied in order to obtain practical and functional applications. Thus the sensor analyses require integration of different libraries in order to perform their tasks. These libraries are generally developed for processing of signal or multimedia resources such as video, audio and image and they need to be adapted in order to be utilized.
- Develop/adopt necessary tools for sharing the results: after the results are deduced, or the sensor values are supposed to be delivered to other environments, the data transfer from device to device may introduce some great difficulties. Machine to machine (M2M) communication is usually an essential requirement of this type of application.

These challenges are valid most of the time, for most of the development of sensor based vehicular mobile application that has the aim of creating a smart application.

1.2 Mobile Sensor Framework

A framework is developed in the scope of this study in order to overcome the challenges of developing sensor-based vehicular mobile pervasive applications. The aim is to enable easy, fast and flexible implementations of vehicular mobile applications using this framework. Although any kind of sensor-based application implementation may be developed, due to the dynamic environment of vehicles, the main goals are to enable real-time sensing of contextual information, multimodal sensor analysis for signal and image processing activities and effective, fast, easy sharing of results in social environments and cloud-based solutions.

The “Multimodal Sensor Analysis Framework for Vehicular Mobile Applications” is developed for mobile devices, utilizing their already integrated diverse sensing and computing capabilities, but a server component is also present, for central access and information sharing purposes. The framework is developed on the popular mobile platform Android, which provides a flexible software development environment. Interfaces are defined in the framework and various systems and libraries are integrated in order to provide diverse functionalities that may be implemented easily and effectively. The developers may utilize several sensors and implement different analyzers based on pre-computed values of sensors in order to develop multimodal sensor applications. They may also extend already defined interfaces and utilize the libraries of the framework.

1.3 Document Organization

This document is organized as follows: the second chapter includes the literature survey related to this study. We first start with the investigation of mobile sensor frameworks, similar studies and applications in regard to smart vehicle concept. We then review the

methods of specific sensor analyzing systems, especially focus on road sensor values analyzing. Finally, we focus on obstacle detection based on image processing techniques using a monocular camera system. There are many object detection studies on binocular camera systems, but these studies are out of the scope of this document. The conceptual design of the framework is described in detail in Chapter 3. This chapter starts with the aim of the study and the fundamental concepts related to the study are also explained in this chapter. Then the design of the framework is explained with system architecture and different components of the framework.

Chapter 4 includes the development details of the sample implements using the framework. The chapter start with topics related to the implementation and includes inertial sensor readings, mobile operating system interfaces and SDK, object detection algorithms. Then the development environment and the developed modules are explained in detail. The server application, the messaging between clients and server, are also discussed in this chapter.An experimental scenario is prepared for this study and is explained in Chapter 5. This includes the setup environment, the components of the hardware systems, the expected events to be detected and the outcomes. The scenario is run by different users and the results are documented. A set of sensor values are collected during the run of the scenario.

Finally, Chapter 6 is prepared to resume the work done and includes the results of the study. The experimental scenario results are especially important in regard to demonstrate the abilities of the framework and exhibit potential usage for sensor based vehicular applications.

CHAPTER 2

LITERATURE SURVEY

The literature survey chapter is composed of two main sections where different parts of the system are analyzed. The first section is about the collection and recording of sensors data and possible usages of this data for similar purpose. This section includes sensor frameworks developed to collect and utilize sensor values for different purposes. Specific methods focused on road analysis which are developed to measure the "roughness" of the road by profiling the road surface using different techniques are also examined in this section. This procedure is called "Road Profiling" and most profiling methods as well as different use-cases of the methods are explained. Specific studies about the topic as well as commercial products are also provided in the same section.

The second part of literature survey is about computer vision domain and targets especially studies of different in-vehicle video analysis applications in traffic. The process of detecting different objects (road, lines, signs, vehicles, pedestrians etc…) and obstacles (box, pedestrian, trees, etc…) on the road are analyzed in this section.

2.1 Sensor Analysis

The rapid development of smart phones equipped with several sensors has commercially reached a large interest. On the other hand, this development has also proliferated hardware dependent academic studies. Today's typical smart phones provide most of the following sensors:

- Global Positioning System
- Accelerometer
- Digital Compass
- Gyroscope
- Ambient Light
- Proximity Sensor

Camera and microphone should also be listed as sensing devices that are ordinarily provided.

Academic and commercial works make use of these sensors for several different application areas to solve problems and ease our everyday lives. Among other various

usages, smart vehicle is one of the prominent application areas of this domain. The detection of road quality in highways, namely “road profiling” is one possible usage of such technology. This concept also includes the detection of potholes, cracks, expansion joints, manholes and other irregularities of the road surface. Another application area is based on recording all the “events” in a vehicle while driving, in order to record data and use it in case of an incident or accident. The usage of event recording is analogous to “black boxes” in aircrafts. This type of recording may also be used as a “travel diary” and record visited places’ location, images/videos as well as other sensor values.

2.1.1 Sensor Frameworks

2.1.1.1 Social fMRI

Friends and Family (FunF) [30] is a sensor framework that enables usage of various sensors of Android mobile devices to collect data and use it for social and behavioral analysis. The framework is based on a study called “social fMRI” [31], which use the smart phones to measure and understand social mechanisms in the real world. A mobile phone sensing platform is being developed within the study which constituted the basics of FunF.

Social fMRI aims to sense and explore (analogous to MRI in medical) the effects of social systems through use of mobile phones, credit cards, social media and telecommunications. The system is being developed as a “ubiquitous social observatory” and the methodology it uses is as follows: users are being monitored by various means such as sensors in mobile devices, call logs, social network activities, surveys, spending (through receipts and credit card statements) etc.. The system then introduces actuators and interventions to the social system and explores the effects by various means of sensing capabilities.

The main contributions of the study may be listed as:

- A dataset of sensor values collection from 25 users. The dataset includes data about location, accelerometer, proximity of other devices (via Bluetooth), phone calls, application list (installed, running), file system information
- The FunF platform which enables the usage of “probes” to collect mobile device sensor data. Mechanisms for feedback are also implemented. The platform is deployed to 130 adult users and used for more than a year.
- A newly proposed social mechanism, called Peer-Reward, which aims to reward the user by measuring the performance of his/her peers.

The newly proposed social mechanism is being tested in a setup of fitness-centered social interaction scenario. The participants’ mobile devices are installed with a sensing application and their activity levels have been measured and classified from 1 to 3 (1 inactive, 3 very active). The activities are determined according to a simple statistical analyzer which used the accelerometer values.

The result of the study shows that real world physical activities are significantly affected by the social components and the proposed Peer-Reward mechanism helps to increase the efficiency of resources invested in reward-based systems.

2.1.1.2 FunF

As stated before, FunF is built in Social fMRI study, and currently provided as an open source framework. This framework basically provides a set of functionalities with openness and reusability in mind. Several data including sensor values of mobile devices may be gathered using the underlying infrastructure.

The probes are self-contained unit of collectors, which are responsible for collecting and providing values for a specific type of information. Some of these probes are: GPS, Cell Tower ID, Contacts, Call log, Running Apps, Music/Video/Image file scan, Battery status, Bluetooth etc… The framework provides built-in probes as well as interfaces to build custom probes.

One of the main concerns of the framework is to protect privacy while collecting the information. The human readable sensitive information in sensors such as real identities, names of contacts, phone numbers, text messages etc… are hashed within the probes and stored only in this format.

The platform is provided in different formats for different usages. There is a readily usable application, in which the sensor probes are selected and values are uploaded to the selected destination. Developers may utilize the API provided by the platform and develop their own mobile applications. The sensor reading, uploading and configuration is handled by the platform, whereas other implementations are performed by the developers. For a core research study, the platform's open source code may be utilized and low level development may be performed on all parts of the framework.

2.1.1.3 Open Data Kit - ODK

OpenDataKit [32] is an open-source set of tools to collect and aggregate sensor values with the aim of building surveys in domains such as socio-economics and health. The toolset is based on creating survey forms and collecting data are based on the configuration made on this forms. The web site of the framework provides interfaces to create forms. The user defines the required information that will be collected during the survey using these interfaces and saves the forms to a personal account. Then the mobile device connects with this account and accesses the forms. Then, the manual process of survey starts upon selection of a form, and the user fills the information that has been requested by the form. A sample built-in form is a geo-tagger, in which a picture is taken manually, and the application tags with the location information automatically. Then this information is uploaded to the server application.

The main focus of the framework is to develop survey forms and provide availability to these forms via mobile applications. For this reason, the sensor data could not be processed but only collected and aggregated on the server side. Thus, this framework is not usable for real-time vehicular mobile applications.

2.1.1.4 Aware Framework

Another framework is Android Mobile Context Instrumentation Framework (AWARE) [35] which provides logging and sharing sensor values of Android platform. The framework provides a client and a server application.

Using the client application, developers are able to implement new “Context Sensors” to create custom sensor logging mechanisms. The collected data is then uploaded to server application where developers are able to analyze the data and/or configure client applications. Several sensors are accessible through the framework such as accelerometer, GPS, temperature, Wi-Fi, barometer, light etc… The developer is able to configure the debug state (debug on, debug off), the logcat tag string and the device id (UDID) which is set initially on installation by default. The client application may operate on standalone mode in which the sensor data is logged locally. On the other hand, the client may also work cooperatively with the server application in which it may send the sensor values either via message using MQTT protocol or sending periodic bulk of sensor data via web services. The messaging type of data exchange may be performed either with the server applications or also with other framework clients.

The framework introduces the concept of “Add-ons” which provide high level contextual data deduced from one or multiple sensors. Currently, there are five defined add-ons:

- Mode of Transportation: the current mode of the device as standing/still, walking, running, biking or driving a car.
- Pedometer: detection of the user steps while walking
- Polar HRM: detection of user’s heart rate with an external heart rate measurement sensor, communicating via bluetooth
- Screen Orientation: detection of the orientation of the screen
- Social: detection of online contacts on social networks

The add-ons provide an on-off switch and a settings page.

The framework is not usable for real-time sensing and signal or multimedia processing but provides only collection of sensor values for offline analyzing via logging of data. The documentation of the framework is poor and the runnable applications are currently not available through the web site.

2.1.2 Road Profiling

Road profiling is the technique to measure the roughness of the road surface. Devices that operate to make such measurements are called profilometers [1] and several different profilometers are designed [2]. The measurement units of these devices are usually expressed in terms of International Roughness Index (IRI) which is the “road roughness index most commonly used worldwide for evaluating and managing road systems” [3] and which has unit (in/mi, m/km) as the accumulated slope of the road.

Although specific hardware solutions are developed for road profiling ([4],[5]), these solutions have major disadvantages such as very costly deployment, maintenance and operation, limited measurement area due to vehicle speed etc…

On the other hand, solutions using sensors on smart phones are considered to achieve cost effective good-enough results. Although the measurement quality is degraded with respect to specific hardware solutions, this type of application is emerging rapidly due to inexpensive hardware and ease of deployment. This advantage leads also to collect an extensive amount of measurement from public usage (crowd sourcing) of the system.

2.1.3 Pothole Detection

2.1.3.1 Pothole Patrol

"The Pothole Patrol" [6] is a study that is focused on mobile sensing to detect road surface condition and report it to a central repository. The system uses only the accelerometer and GPS sensors in order to detect road anomalies and is deployed to a specific platform with Linux operating system and sensors attached externally. Raw sensor values are collected (location, speed, heading, 3-axis acceleration) and the detector is run on the device itself in real time. The Figure 1 describes the detector algorithm of the system which consists of simple machine-learning algorithms. The (vertical) z-axis and sideways (x-axis) acceleration values are processed with the speed of the vehicle to obtain pothole detections and five filters are executed sequentially. The successful detections are sent to central server which "combines these to produce a set of "road anomalies" including potholes and other rough road conditions" [6].

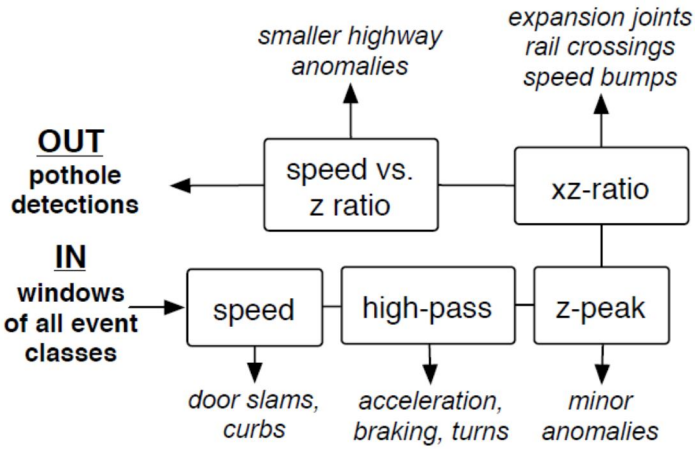


Figure 1 The Pothole Patrol detection algorithm (drawing [6])

The system is deployed to 7 cabs (taxi) for testing and they have traveled a total of 9730 km with distinct 2492 km and identified 48 pothole or other road anomalies with 90% confidence.

2.1.3.2 Real Time Pothole Detection using Android Smartphones with Accelerometers

"Real Time Pothole Detection" system [9] is using Android smart phones to detect potholes in real time and the study is focused on successful detection of the road infrastructure. Different techniques on the sensor values collected from accelerometers are employed in this study, namely: Z-Thresh, Z-Diff, Stdev(Z) and G-Zero.

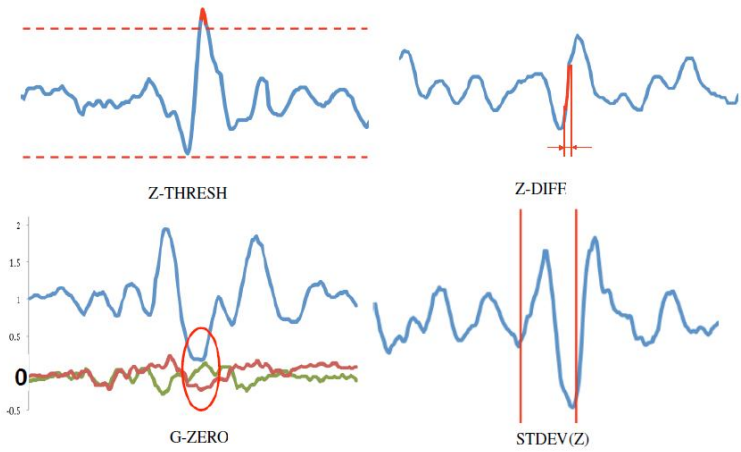


Figure 2 Real Time Pothole Detection techniques (drawings [9])

The evaluation of the system is based on the comparison of the measurements of the system for different algorithms relative to ground truth measurements, collected by walking on a 4.4km long test track region. Different types of road irregularities such as large potholes, small potholes, pothole clusters, gaps and drain pits are included in the evaluation.

Class	Z-THRESH	Z-DIFF	STDEV(Z)	G-ZERO
Large potholes	3 (100%)	3 (100%)	3 (100%)	3 (100%)
Small potholes	15 (83%)	16 (89%)	16 (89%)	14 (78%)
Pothole clusters	25 (83%)	27 (90%)	27 (90%)	27 (90%)
Gaps	31 (78%)	36 (90%)	30 (75%)	27 (68%)
Drain pits	10 (59%)	17 (100%)	11 (65%)	8 (47%)
Total	**84 (78%)**	**99 (92%)**	**87 (81%)**	**79 (73%)**

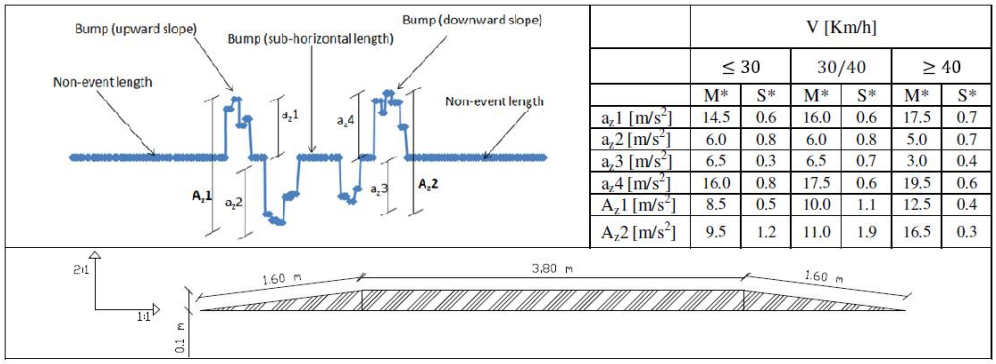
Figure 3 True positive rates of the four used algorithms (table [9])

Different true positive values are obtained for different algorithms with an average of 81%. The study conclusion foresees a future work on combinations of algorithms as well as an implementation of a new functionality for self-calibration of the system.

2.1.3.3 A mobile application for road surface quality control: UNIquALroad

UNIquALroad study [10] is aiming to monitor the quality of the road by detecting potholes and bumps using accelerometer sensor data of regular mobile devices. The system consists of a standalone mobile application that measures accelerometer values and applies simple filters in sequence. The algorithm uses the acceleration on z axis to detect vehicle's perpendicular movement relative to the ground, and uses other axis values for accelerometer reorientation using Euler angles. The reorientation process is performed while stationary using XYZ sequence of Euler angles.

The road anomalies are first manually analyzed and characteristics of accelerometers values are identified. Figure 4 shows the accelerometer z-axis change over time for a typical stone bump under various vehicle speeds.



	V [Km/h]					
	≤ 30		30/40		≥ 40	
	M*	S*	M*	S*	M*	S*
a_z1 [m/s^2]	14.5	0.6	16.0	0.6	17.5	0.7
a_z2 [m/s^2]	6.0	0.8	6.0	0.8	5.0	0.7
a_z3 [m/s^2]	6.5	0.3	6.5	0.7	3.0	0.4
a_z4 [m/s^2]	16.0	0.8	17.5	0.6	19.5	0.6
A_z1 [m/s^2]	8.5	0.5	10.0	1.1	12.5	0.4
A_z2 [m/s^2]	9.5	1.2	11.0	1.9	16.5	0.3

Figure 4 Stone Bump Peak Accelerometer Values (M*: average value; S*: standard deviation) (drawing [10])

Considering the accelerometer z-axis change during a bump traversal, the implemented algorithm applies the following filters sequentially: Low-Frequency Filter (LFF), Speed Filter (SP) and Small Peaks Filter (SPF).

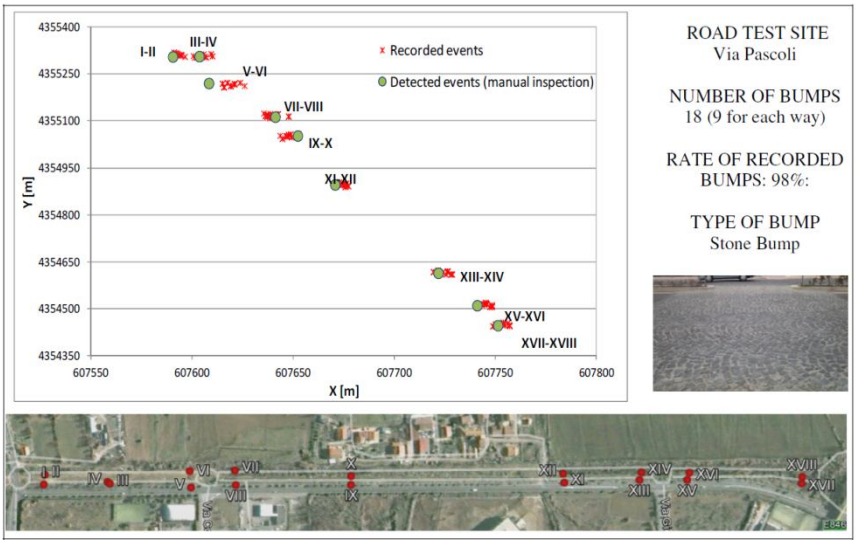


Figure 5 Test Results for Via Pascoli test site (drawing [10])

The developed application is tested on two different test sites (Unical and Via Pascoli) for an overall 8 km distance. The conducted tests are performed with a driving speed between 25 km/h and 40 km/h. Two different accelerometer frequencies were used: 5 Hz and 100 Hz.

The evaluation of the study shows that a reasonable level of bump detection rates over 90% true positive are achieved with the application. However the developed pothole detection method performed poor with recorded potholes as low as 65% localization.

2.1.3.4 Others

Other studies on pothole detection are Microsoft's Nericell [7], TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones [8], Speed Breaker Early Warning System [11], Automatic Road Anomaly Detection Using Smart Mobile

Device and Road Condition Monitoring Using On-board Three axis Accelerometer and GPS Sensor [13].

2.1.4 Vehicle Event Data Recording

Vehicle Event Data Recorder (VDR) applications track the sensor values collected by a device mounted on a vehicle. Although the tracking may be performed for different purposes, the main target is only to record the sensor values and utilize them in case of an accident, an undesired incident or just for sharing in online social networks. Thus the main goal of these types of applications is only record relevant sensor data such as accelerometer for the motion of the vehicle, GPS for the location and the speed of the vehicle and the camera for recording the scene. For this reason, these applications are also called “car black box” (in relation to black boxes used in airplanes), dash cam or Digital Video Recorder (DVR).

2.1.4.1 DailyRoads Voyager

DailyRoads Voyager [35] is an Android application which is used while driving on road. The aim is to record the video and audio while driving and save the video automatically upon a detection of a hazardous event such as an accident. This application is analogous to black-boxes in planes and serves as an evidence source in case of a disagreement.

Figure 6 DailyRoads Voyager main screen ([35])

The application records sensor values such as accelerometer, GPS and video camera and runs algorithms to automatically identify “important” events such as incidents or accidents. Accelerometer and speed sensor values are tracked and significant changes are identified, in which case the video and other sensor values are saved. Other unimportant video files and sensor data are wiped-out automatically. The application data may be uploaded to a web server and further analysis may be performed on this data.

2.1.4.2 Safe Start

Safe Start [37] is also a video recording application developed for Android platform and records sensor values such as accelerometer, GPS (location and speed) and video using the camera of the mobile device and classifies itself as an “Event Data Recorder”. The application performs recording of sensor values and detects hazardous events such as accidents.

Additional to similar solutions, the application support ODB2 hardware interface integration. Using the Bluetooth connection, the application may connect to the engine of the vehicle via ODB2 interface and collect info such as speed, RPM, load, throttle position, intake air temperature, water temperature, fuel pressure etc…

2.1.5 Social Vehicular Applications

2.1.5.1 SVN: Social Vehicle Navigation

Current navigations systems (either as a separate navigation device or an application on smartphones) provide advanced navigation features such as real-time traffic flow, multiple routing options based on different pre-defined criteria, personalized routing etc… Real-time traffic data may be provided by gathering GPS-enabled vehicle’s current speeds combined with past traffic flow information. However, the users require more information about the navigation systems day-by-day. The road conditions, specific reason of a traffic jam or the possible alternative routes are some of the information that are difficult to automatically extract by systems and applications.

The traffic condition may change within a short time and these requirements may alter swiftly. Besides providing real-time traffic information, it is beneficiary to provide the experiences of vehicles’ ahead to the ones which are behind in a crowd-based sensing manner also. In this context, the study of Sha et al [34] tries to develop a social navigation system, which captures the experience of drivers and shares them with other drivers in order to “calculate personalized routes”. The system is based on “voice tweets” (tweets that include a voice record file) produced by drivers that include the current status of the traffic they are in and sent to a central server. The server organizes the voice tweets according to their sending location and drops the old tweets as newer ones are received. Thus, “tweet digests” are created for specific regions and sent to drivers in these regions. The drivers listen to the digest one by one and provide audio inputs by simple commands such as “avoid” or “choose” and the navigation system selects the road that contains the location tagged in the voice tweet.

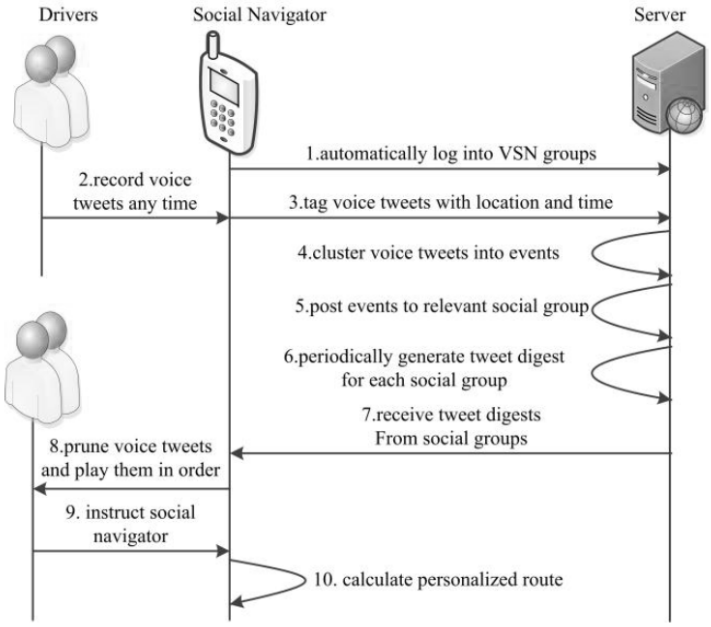


Figure 7 SVN Interactions [34]

The sequence diagram in Figure 7 shows the interactions of the driver with the social navigator and interaction between the navigator and the server. In this system, the regions are defined either as a “destination group” with a geo-location and a radius defining the range of the group or a “road segment group” for a main road, separated by different exits. The server application provides user interfaces via a portal application and the drivers are able to define their own groups and each driver may register to any number of groups. The mobile application is implemented as an Android application and on the route, the driver touch the screen and starts to record a voice record for a maximum of 15 seconds and when finished, the record is automatically tagged with time and location and sent to the server. Since the tagging of location is performed automatically, the driver should make the recording as near to the event location as possible.

After the record is sent to the server, the application clusters the records according to the groups defined, and time. Older records about the same location in one group are purged; other records for the same group remain clustered. Thus, each group maintains the most current voice records on each location informed. The mobile application downloads the voice records as soon as a new “digest” is created on the server side for the registered groups and starts playing. The driver listens to the records in the digest one by one, and provides commands to the system by audio inputs. The driver’s commands are in the form of “avoid” or “choose” after listening each record, or “avoid route 1 and 3; choose route 4” after listening all the records. The navigation system then recalculates the best route according to these decisions.

For the bandwidth utilization of the system, the authors calculated the total data transferred for a driver who receives 20 a day (each of 15 seconds) with a size of 5-8 Kbytes each as 6.7 Mbytes for a month.

2.1.5.2 Waze: Community based mapping, traffic & navigation app

Waze is an open source application that collects/provides real-time mapping, traffic and navigation information from/to its users. It is a crowd-sourcing based free navigation application that supports various features related to traffic and road hazards. Currently, the application is provided for both iOS and Android devices and recently Google acquired the company [43].

The system is a crowd-sourcing based map editing and navigation application. Users run the application while driving and get navigation data, while at the same time they are also updating the map in the background for the roads they have passed. The application provides manual map editing options such notification of hazard, traffic light, police car, closed road etc… The number of users all around the world is mentioned as 50 million.

2.2 Computer Vision Applications on Road Video Analysis

2.2.1 Object Detection

2.2.1.1 Towards Detection and Tracking of On-Road Objects

In the study of "Towards Detection and Tracking of On-Road Objects" [38], Goecke et al. developed an object detection system that works on a monocular camera while driving on road. The object detection algorithm operates in two steps. The first step eliminates the pixels of the image above the horizon line, dynamically, so that the processing time of the object detection is reduced dramatically. The second step includes the detection of on-road objects by subtracting pixels from the background scene. The system is developed based on "cues" and several cues are calculated and combined for the final result.

For the subtraction of the background, the road surface is detected and marked as background as an initial cue. To do so, the color space of the frame is converted to HSI color space and a simple thresholding is applied by subtraction current hue and saturation from mean hue and saturation and comparing to their standard deviations with a factor. This method assumes that there is not a heavy traffic and the road surface is visible in most of the time. Otherwise, the thresholding will not achieve good results and the system final detection will be poor. For the second cue, the Shannon Entropy is applied over rectangular areas. The calculated entropy value is compared with a threshold in order to identify it as a background or foreground object. These cues are then evaluated based on a voting mechanism and pixels marked as foreground for both cues are assumed to be part of foreground objects.

In the second step, Response Binning algorithm (an extension of AdaBoost and RealBoost approaches introduced by Viola and Jones) is applied in order to detect the cars in the foreground image parts. A sample set of 2037 positive and 5000 negative images is utilized for the training of the algorithms.

The study concludes with mentioning the results as partly successful since the system works on low-traffic conditions. It will detect the cars when they are disjoint but fails when occlusion occurs. The processing speed may achieve as high as 10 Hz. No experimental measurements are provided within the study.

2.2.2 Obstacle Detection

2.2.2.1 Pothole detection in asphalt pavement images

In the scope of obstacle detection, the study of "Pothole detection in asphalt pavement images" [38] performs a shape-based extraction and detection of the potholes in roads. The system is simulated using a robot (Figure 8) equipped with a high-speed camera. The processing is performed offline, after collecting the image using the robot.

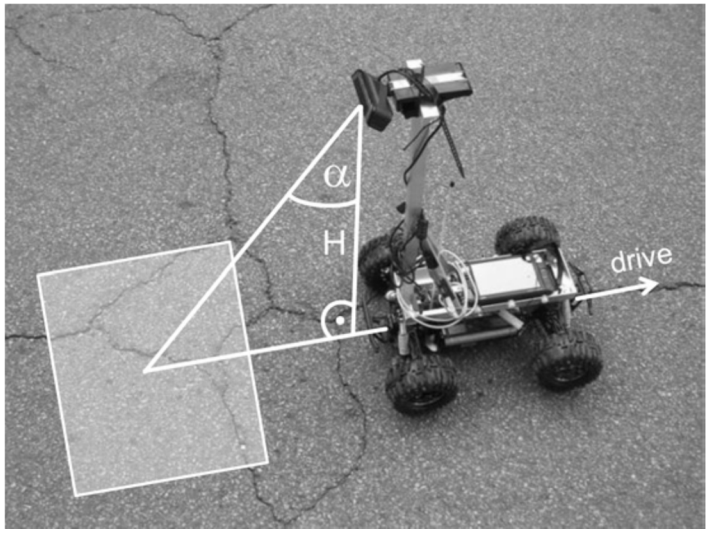


Figure 8 - The robot vehicle used in the study in order to collect asphalt images [38]

The system locates and identifies the potholes using different combinations of a set of image processing techniques. The first step is the image segmentation in which a histogram shape-based algorithm is employed in order to convert color images to gray-scale frames and expose the darker parts constituting the potholes. The second step is extracting the shape of the pothole for which the study assumes approximately all potholes have an elliptical shape (due to perspective view) and eliminates other types of potholes (such as cracks, lines etc…). In order to identify the pothole candidates, the shape extraction algorithms calculates some parameters as length of the major axis, position of the centroid and orientation angle in order to be utilized in finding an elliptical shape. Once the region of the candidate potholes are identified, the textures of the inside and outside of the region are compared based on the standard deviation of gray-level intensity values as a statistical measure, in order to classify false-positives and true-positives.

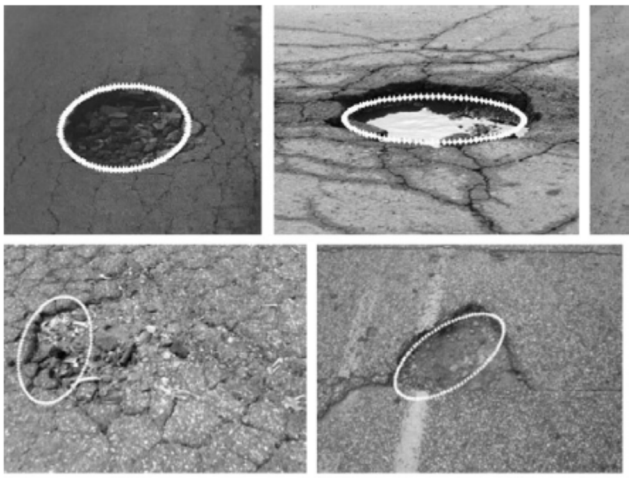


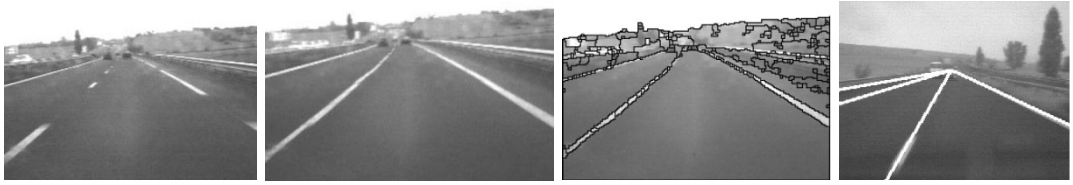
Figure 9 - Some of the pothole detection results of the study ([38])

The study output is tested on a set of images containing potholes and the results show that detection ratios reach to 82% for precision and 86% for recall values. The study relies on the daylight illumination effects which create shadows on potholes and will perform poorly when there is no sunlight or with direct sunlight from top.

2.2.2.2 Road segmentation and obstacle detection by a fast watershed transformation

Beucher et al. [40] developed a system to detect object on road using a monocular camera. The system operates in two steps. In the first step, the road segmentation is performed and the pixels of the frame corresponding to the road are extracted. In the second step, the objects on the remaining parts are identified. The study uses a special hardware for onboard image processing.

For the road segmentation, first a line detection algorithm is run. To do this, a morphological temporal filter is applied on successive images in a time window and an edge detector is applied on the resulting temporal dilation. With this process, the regions of interest that would probably contain the objects on the road are identified. Moreover, identifying the lanes provides a means to compute the estimated distance to the objects ahead of the vehicle. The algorithm first runs with a larger size of temporal dilation for the detection of the objects then continues the process with a tracking algorithm utilizing a smaller number of frames in order to speed up the detections. The detection of lines is summarized in Figure 10.



(a) (b) (c) (d)

Figure 10 Line detection Algorithm: (a) raw frame (b) temporal dilation (c) edge detector (d) line drawing

As a second step, the object detection algorithm is applied. To find the objects in the extracted regions, two assumptions are considered: (1) there is a region generated by the object which is darker than the road and (2) the vehicles are rectangular shaped having horizontal and vertical contours. The region of interest is processed in this step and the regions having darker color are further filtered using a morphological transform called h-minima. The remaining regions are further filtered by checking the dimensions of the object and whether it is on the road or not.

(a) (b)

Figure 11 Object Detection on the extracted region of interest: (a) darker regions are identified and (b) the objects are filtered according to their dimensions

The study did not mention about the rate of success of the algorithms but claims to run all the algorithms for each scene in less than 200 ms.

2.2.2.3 Motion-based Obstacle Detection and Tracking for Car Driving Assistance

As part of Carsense European project [41], this study [42] focuses on detection and tracking of obstacles on the road from a front camera mounted on a vehicle. An image processing based solution is presented in this study in order to detect obstacles which are at most 50 meters ahead of the car, for which the system should operate in low vehicular speeds. The different types of obstacle to be detected are predefined as vehicle (car, truck), motor-bike, bicycle or pedestrian.

First, the dominant image motion due to the camera motion is computed and a weight map is extracted which is used for the detection of the foreground objects. Each pixel that is not conforming to the weighting map is considered as a part of an obstacle thus processed as a foreground object. After merging foreground pixels and morphological operations, the obstacles' bounding area are easily identified. In order to minimize the false positive detection rate, the bounding areas are tracked over successive frames and an evaluation of the motion consistency in the intersection of overlapping regions in consecutive frames is performed for the tracking. Using the motion model, the detected obstacles' time-to-collision values are also calculated using the trajectories of the obstacles.

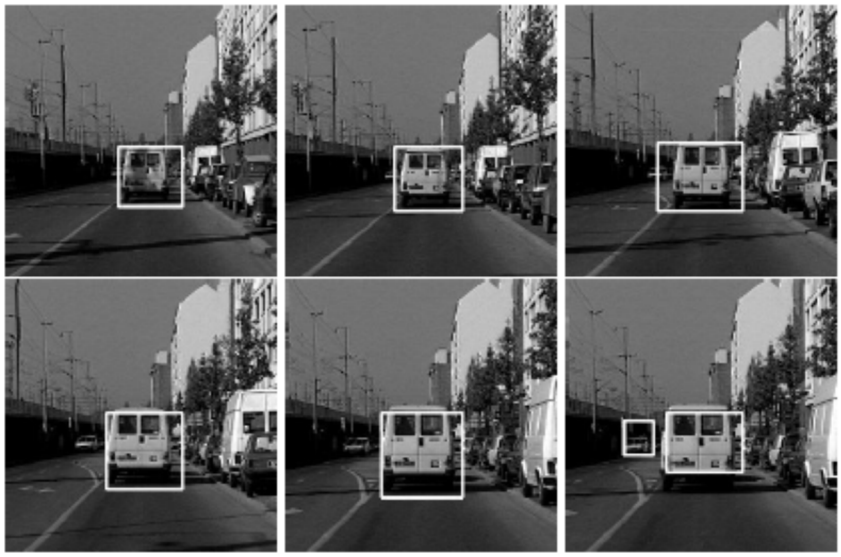


Figure 12 Object detection and tracking with a camera mounted on a car [42]

The system operates with images having dimensions 512x512 and the results show that the object detections are performed with a speed of 2.5 frames per second.

CHAPTER 3

MULTIMODAL SENSOR ANALYSIS CONCEPTUAL DESIGN

3.1 Aim of the Study

The convergence of mobile technologies enabled the smart phones to provide various usages due to their large and diverse sensing capabilities. The location and motion sensors as well as camera are the most commonly utilized sensors to realize functional and pervasive mobile applications. Applications using sensors perform their tasks in three main steps which are collecting, analyzing and acting.

Figure 13 Main steps of sensor based applications

The review of literature shows that sensor based mobile application is an active research area and several studies are being conducted. There are many application domains utilizing the sensors of mobile devices and vehicular applications are promising because they are handy, practical and usually do not require an expenditure or complex setup of a special hardware.

Frameworks have been developed and actively used by the community to ease the usage of sensors on mobile devices, and develop functional applications. “Social fMRI” [31] study and FunF platform [30] use the smart phones to measure and understand social mechanisms in the real world. These platforms enables access to all kind of sensors, collect data with configurable settings and share the measurements. However, the platform is being developed for social analysis in mind, thus does not require and provide real-time sensing capabilities. Moreover, the focus of this study is on developing vehicular sensing applications; however FunF does not have abilities such as integration of signal and image processing libraries. OpenDataKit [32] is an open-source set of tools to collect and aggregate sensor values with the aim of building surveys in domains such as socio-economics and health. The toolset is based on creating survey forms and

collecting data are based on the configuration made on these forms. The sensors could not be processed but only collected and aggregated.

Mobile applications are developed for on road analysis including pothole/bump detection, road monitoring, road profiling, vehicle event data recording, obstacle or pedestrian detection and avoidance etc… Most of the studies spend extensive effort on the collection and analysis of sensor data and an extra effort for sharing the results although their main concern usually is only to devise a method that improves the state of the art. These extra efforts increase further when there is no familiarity with the development environment of sensing devices and/or data transfer technologies.

Moreover, the previous studies focused on a specific sensor usage, either camera or accelerometer. The camera focused studies usually realized specific systems which are developed with special setups, by employing sensors, cameras, IR cameras or other high cost equipment. Some of these systems' aim is to extract the profile of the road and use this profiling information for the maintenance of the road, investigating possible improvements in road construction or modifying the vehicular traffic flow plans. These are usually case specific solutions for problems of municipality, construction firms or similar organizations. On the other part, some systems are focused on binocular cameras, and employed these devices in the detection of particular objects such as pedestrians, traffic signs, road lines, other vehicles etc…

All of these systems are generally providing much detailed and complete solutions to specific cases; however they usually require much precise sensor data thus the high quality equipment is extremely expensive. Also some systems usually need a special setup for specific vehicles. Moreover, these systems are costly in terms of operation and maintenance, too. For this reason, this type of systems does not provide a solution to all kinds of drivers in traffic and are impractical for daily usage.

Systems that make use of only accelerometer sensors have also been developed to detect potholes, bumps, cracks on roads. Some of the systems work standalone, just for collection and classification of road data for offline operation. Some others are connected to each other via a central application; however they provide only the location and possibly the type of the incident as pothole, bump, crack etc. They provide only statistical outputs while missing the image of the scene or the specific cause of an obstacle in order to visualize the case. Some systems are operating offline and cannot share information.

Within the same context, Vehicle Event Data Recording systems have also been developed in order to record the sensor values, including video from the camera, and perform analysis on accelerometer data (to detect collusions for example). This type of application is useful for post investigation purposes such as using the records as evidence of traffic accidents or reporting illegal traffic behavior to authorities. Another possible usage of these systems is to create a personal journey of driving activities. This type of applications is usually commercial applications that do not implement complex analysis methods and used just for recording sensor values.

The aim of this study is to develop a framework that would enable easy and flexible usage of mobile device sensors, implementation of analyzers on these sensor values and means of sharing the results deduced from the analysis.

This study basically focuses on the challenges of developing sensor-based vehicular mobile pervasive applications and aims to construct a framework to enable easy, fast and flexible implementations of such applications. Although any kind of sensor-based application implementation may be developed on this framework, the main goals are to enable:

- real-time sensing
- signal and multimedia processing
- effective, fast, easy sharing of results

The framework's base component is mobile devices, due to their already integrated diverse sensing and computing capabilities, but a server component is also present, for central access and information sharing purposes. The framework efficiency is demonstrated best with a relevant use case. For this reason, a sample implementation of road analysis application has also been realized within this study.

There is no limit on the usage of this framework; anyone who wants to develop sensor based application may utilize the development framework.

3.2 Fundamental Concepts

Before diving into the details of the conceptual design, mentioning about the fundamental concepts used in this study would clarify much better the system and the components of the system. For this reason, this chapter is divided into two sections, in which concepts about the mobile device are listed first, and concepts about the server side are listed next.

3.2.1 Mobile Device Software Development Kit

The framework is developed on Android SDK, because it provides access to any kind of sensor with higher permissions than iOS SDK. Android SDK includes a sensor manager that this framework utilizes. However the sensor framework of the SDK requires a good understanding of sensor usage, registrations as well as data collection. This framework's aim is to provide an abstraction over these interfaces and to provide a clean and simple access to sensor readings.

3.2.2 Mobile Device Sensors

This section explains the following mobile device sensors: GPS, Magnetometer, Accelerometer and Camera. Each sensor has its own constraints related to the study. These constraints are also mentioned in the subsections.

3.2.2.1 Global Positioning System

GPS sensor provides two kinds of information. The first one is the location of the device. The location information may be fetched from the sensor each time it provides an update of the data. The location data consists of latitude, longitude and altitude information of the position of the device. The accuracy of location information is closely related to the device quality as well as external conditions such as atmospheric effects, nearby tall buildings, or any other device that would interfere with GPS signals. The location accuracy is 3 meters or better for most of the GPS receivers [14]. Another limitation of GPS device is the time required to start providing location information. After the device is operational, it may take up to 5 minutes before providing the location information. The second information that GPS provides is the movement speed of the device, thus the speed of the vehicle. This information is calculated using the location information for different times, thus is working as a positional speedometer. The speed information may also include inaccuracies and works only near real-time due to location update speed. Although the GPS sensor gives coarse speed information, this study does not require exact and accurate speed information, for which the gathered speed information is quite sufficient to get the rough motion speed.

3.2.2.2 Magnetometer

The magnetometer sensor is a device that measures direction according to Earth's magnetic fields. For this reason, they are also called "digital compass". This sensor is utilized in order to record the direction of the device (thus the direction of the vehicle).

The direction of the vehicle is important because it is crucial to identify the exact location of an event in a two-way road. The GPS sensor values are not accurate enough to identify if a location is on the right side of a two-way road or on the left side. Thus the system should store the direction of the vehicle along with the location information. The server system would handle the necessary direction computations in order to find same direction events.

3.2.2.3 Accelerometer

The accelerometer sensor is a device that measures acceleration. Most modern smart phones have three-axis accelerometers, which provide acceleration information for all 3 axes: x, y and z. While stationary, the sensor provides only the gravity (g = 9.81 m/s2) acceleration on z-axis and 0 for other axis, otherwise it provide the dynamic forces of 3 axis applied to the device. The sampling frequency is selectable for devices providing API's, and most mobile devices yield sampling with frequencies of 50 Hz or even higher. With this level of frequency, it is possible to efficiently measure and detect changes of the device (thus vehicle acceleration) in 3 axes in near real time, which may be used for detection of hazardous events.

The values collected from this sensor reflect only the perceived acceleration and is not the absolute acceleration values. The vehicles tires and suspension system smooth the impact of road hazards, thus the accelerometer sensor perceives only the affected acceleration changes. For the same hazard, different vehicles may produce different

acceleration values affected by their tire types, tire inflation, suspensions of the vehicles and other variables.

Another usage of the accelerometer sensor is to calculate the orientation of the mobile device, in order to get the exact values aligned with the vehicle coordinate system. Using the heading and orientation of the device, it is possible to reorient the device and calculate the normalized accelerometer values with the reorientation, no matter in which position or orientation it is setup. This is quite important for devices to be deployed in vehicles because it is impractical to calibrate the device for a fixed orientation.

Another possibility may be to assume a fixed mounting and fixed orientation of the device, and perform the calculations according to this assumption. However, restricting the device orientation requires a special setup, thus it is not practical to deploy to different vehicles or easy deploy.

3.2.3 Push Notification

Push Notification mechanism is widely used in mobile device environment. The Short Message Service (SMS) of GSM operators is a good example of push notification. In this mechanism, the message in the notification is forced to transmit to the receiving party. The opposite of this mechanism is the 'pull' method in which the receiving party checks the sender if there is a new message for him.

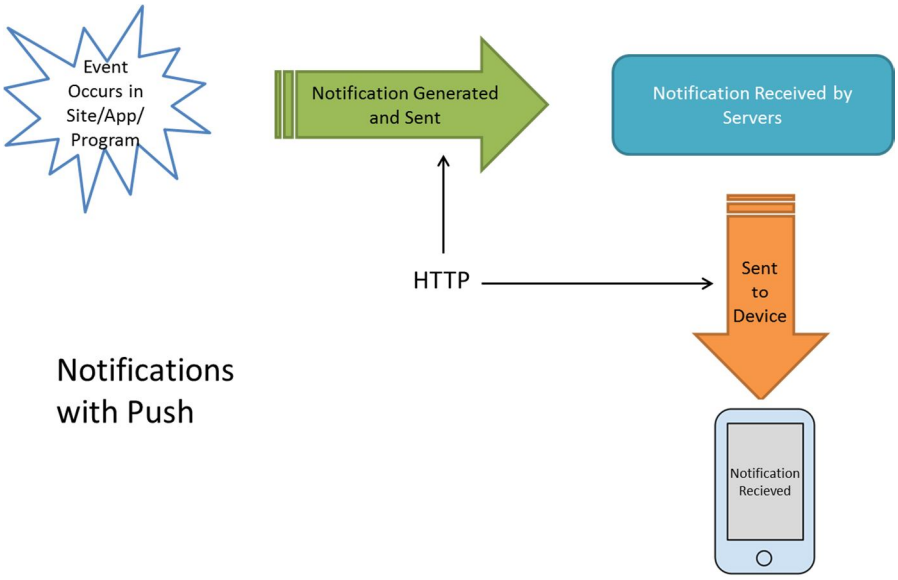


Figure 14 Push Notification for Mobile Devices

There are different implementations of push notification technique in mobile environment such as Google Cloud Messaging for Android (GCM) [15], IBM MQTT [16] and The Deacon Project [17].

This study will implement a push mechanism in order to inform nearby users about the hazardous event that had happened.

3.2.4 Third Party Library Integration

Android SDK provides the usage of external libraries in mobile applications. Java-based libraries are easily ported and integrated into Android projects and utilized without much effort. OpenCV and AChartEngine are such external libraries that may be attached to and utilized.

Android provides also low-level programming, by providing the ability to run native code, written in C or C++ by means of its Native Development Kit (NDK). Android team recommends to use NDK for only self-contained and CPU intensive operations. Libraries such as FFmpeg may be integrated using Android NDK. The communication between java code in application level and C, C++ code in native applications are performed using Java Native Interface (JNI) framework.

3.3 Conceptual Design

The developed framework is designed with the goal of providing necessary tools and methods to easily collect and analyze sensor values and act and share the results flexibly.

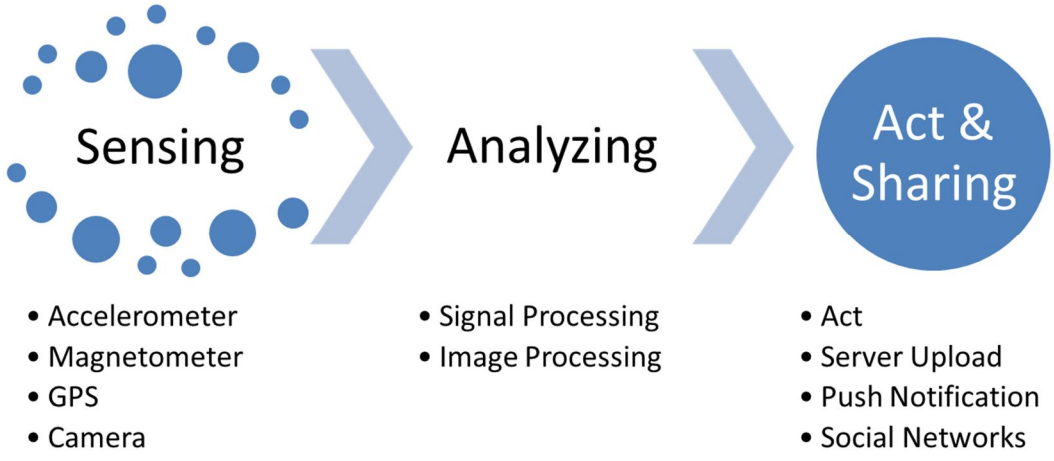


Figure 15 Platform Main Features

Android platform provides different means of implementing application functionalities. The Software Development Kit is the main framework to develop and deploy mobile application on this platform. On the other side, it also provides a level of abstraction over sensor devices with SensorManager. This component in the application framework is responsible of handling sensor registrations, sensor readings and management. On top of this, the multimodal framework provides a higher level abstraction on Android SDK framework and provides interfaces to application level as shown in Figure 16. With this high level abstraction, the applications register, read, utilize and unregister sensors more easily. The sensor data gathered from the multimodal framework is enriched in the

framework. Thus the applications utilizing this framework have access to a complex data instead of raw sensor data. For the regular Android SensorManager interface, the applications should have to deal with synchronizing to other sensor values, enriching with side information and also caching of sensor values in case of abundant data flow (for example with a selection of high frequency of accelerometer sensor). But the multimodal sensor framework handles all of these issues, and provides a fast, reliable and enriched access to sensor values via simple common interfaces.

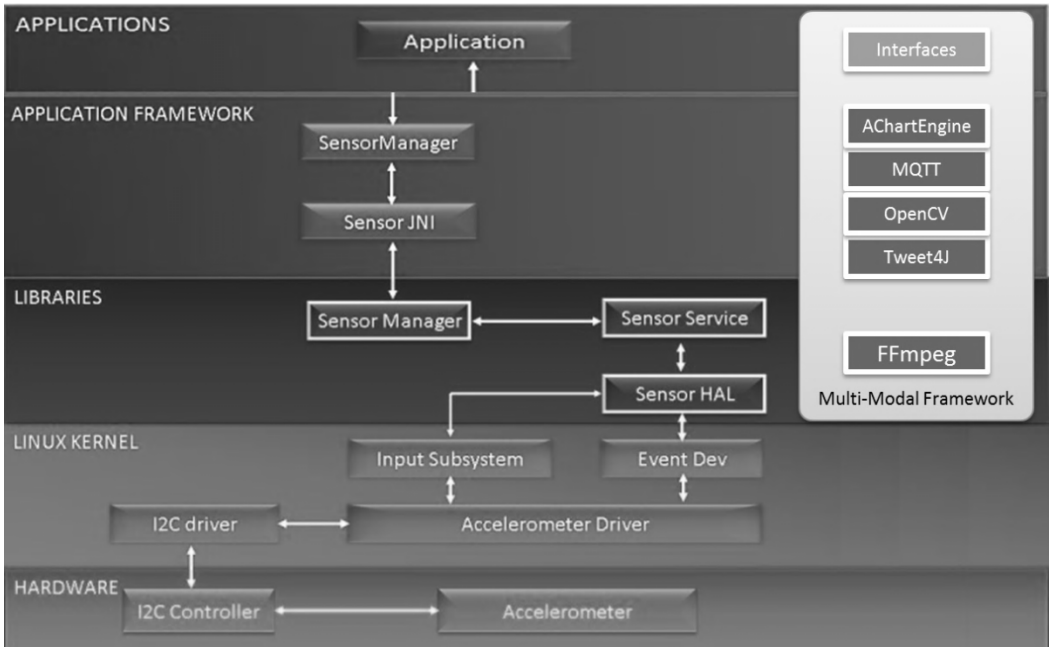


Figure 16 Multimodal Sensor Framework in SDK

On the other hand, Android Native Development Kit (NDK) is also utilized for developing low-level tasks that need extra performance and intense processor utilization. Usually signal and image processing analysis requires high utilization of cpu/memory and Android NDK provides means to implement low level application code in C/C++. It is also commonly utilized to port already developed libraries into Android thereby providing component reuse, as well as platform independence. These low level codes are executed in library space of the operating system. Thus they have direct access to device resources. Some of the external libraries of the multimodal platform are integrated at application framework level. However others (like FFMpeg) require native code execution, thus they are integrated in native side of the platform via Android NDK.

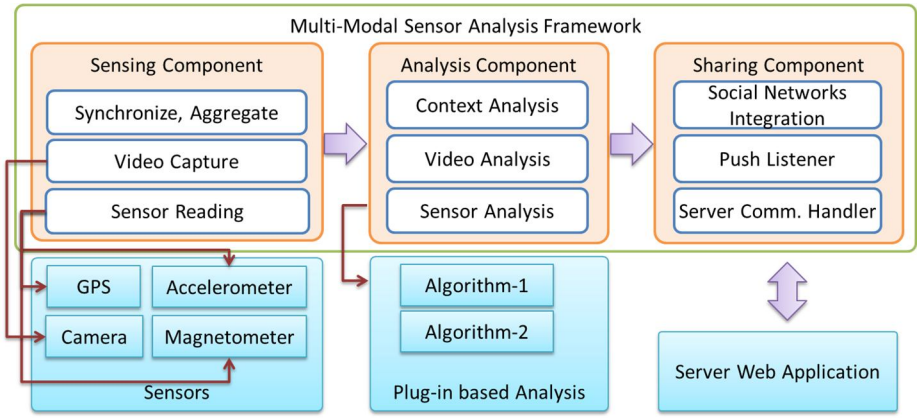


Figure 17 The Framework System Architecture

The framework initially provides a sensing component, which defines an abstraction over sensor readings with the aim of facilitating the accessing, reading, correlating, synchronizing and releasing sensors and their values. The analysis component is integrated with Sensing Component and utilizes the results of the sensor values. It provides interfaces to develop analyzers and perform signal and image processing tasks. The last component is used for sharing the results via social networks or central applications. It supports the push mechanism in order to send messages from server applications to mobile client applications.

All interfaces are explained in the following chapters. For the details of the implementations and specific methods defined in the interfaces, please refer to Appendix.

3.3.1 Sensing Component

The sensing framework in Android is already provided in the SDK with SensorManager interface. The application developer may implement sensing functionalities using this interface and access to raw sensor values for a wide range of sensors. However, one of the problems for reading different sources is the synchronization of these sensor values between different sensors. All sensor values that are processed by this multimodal framework are labeled by a timestamp which is universal across the application. Thus it is possible to find matching sensor values gathered at exactly the same time.

Another problem is the excess generation of sensor values from some sensors such as accelerometer or magnetometer. These sensors provide data in a predefined frequency, which is modifiable by the developer. The developer may set this parameter to UI, NORMAL, GAME or FASTEST mode for accelerometer and magnetometer sensors. The frequencies of these modes are not static, and depend much on the specific device hardware and the operating system version. However the FASTEST mode may reach 100Hz on most of the devices. This makes 100 sensor readings in one second which needs complex algorithms and efficient memory usage. The multimodal framework provides these complex algorithms and defines interfaces to read raw values as well as aggregated, sampled and synchronized values on all sensor readings.

Most of the applications need also to record the sensor readings, either to a file or to a database. This operation also needs careful and efficient implementation of logging mechanism in order to fulfill the requirement of synchronization, aggregation and/or writing to log files or database. This framework provides abstractions by its own to log the sensor readings, both in raw format or in aggregated format.

To provide all of these functionalities, the Sensing Component implements first a threaded sensor collector and processor infrastructure. AcceleroSensorManager and LocationSensorManager are responsible to collect, calculate and inform sensor listeners with the correct type of data (raw, aggregated, or sampled) they have requested. In these Manager classes, the raw sensor values are gathered in a separate thread. This thread (capturing thread) does not include any processing but only timestamps the sensor reading (if it is not present within sensor data) and forwards it to the processor thread. The processor thread implements different processors such as raw sensor collector, sensor synchronizer, sensor aggregator, sensor sampler and sensor logger. The main processor thread forwards the raw sensor value to all processors. These processors are selectable by the developer and they are run with the configuration of the developer. A sensor listener interface (ISensorListener) is implemented within the sensing component, which is utilized by the developer, in order to collect the sensor values. Thus the developer defines only the implementation of this sensor listener interface and decides what to do with the synchronized, aggregated and/or logged sensor values.

The camera sensor may provide a full video file, a partial video provided that start and end positions are specified or it may also provide each frame separately, thus enabling processing of images. The selection should be performed before recording starts, thus the sensor component knows in advance how to start the video recording. The CameraManager handles all camera related tasks such as starting to record a video, setting up the video path and filename, collecting the frame preview, converting to bitmap or saving as a file in sd card. A camera preview interface (ICameraPreviewListener) is also implemented for video recording by sensor listeners, which enables to easily get the requested input from camera sensor.

The FFMpeg library is incorporated in the framework, and all the methods are ready to utilize by developers. For video splitting, this library is utilized through Java Native Interfaces (JNI) calls. The interfaces are defined in “native.c” source file under “jni” directory of the project and they are been accessed through NativeJniCall.java class on the SDK side.

Since the splitting process of a video is a time consuming operation, this process is performed in a separate thread. Thus, a Runnable is defined in NativeJniCall just for processing the splitting requests and it just creates a new thread and calls the jni interface in this thread.

3.3.2 Analysis Component

This component mainly provides interfaces that are compatible with the output of the Sensing Component. Thus, the values collected from the sensors are utilized using the

interfaces defined in this component, which provides easy and fast development & testing of analysis modules.

The analysis component first provides an abstract class named “ASensorAnalyzer” which defines interfaces for easy and effective analysis of collected sensor values. The implementation of these interfaces enables the plugin based structure of the framework, which leads to simple, fast and cleaner development of signal and image processing algorithms.

This abstract class is incorporated into Analysis Component as a plugin-based structure. Thus, it is possible to implement multiple analysis algorithms under multiple instance of this class, which are all run and processed in parallel by the component. All analyzer activities are managed by SensorAnalyzerManager and the implementation are first registered to or created by this class. Then, the lifecycle of each analyzer is monitored and managed till the analyzing is finished.

Each implementation of ASensorAnalyzer includes a list of “ISensorAnalyzerListener” interface implementations. This interface is used in Sharing Component and provides means of acting and/or sharing upon the results of the analysis. This interface is described in detail in 3.3.3.

The analysis component, on the other hand incorporates the use of image processing libraries such as OpenCV [18]. The sensing component provides the frames captured from the camera, and the developer may use OpenCV methods to apply image processing techniques to these captured frames in this component. CvCameraViewListener is utilized and the required methods are implemented. Moreover, the infrastructure includes Ffmpeg [19], the library to manipulate multimedia resources.

AChartEngine [27] is software to create charts, which is also integrated to the framework. DirectShow Java Wrapper (DSJ) [28] is a java abstraction layer around Directshow API[4], and this application is modified and integrated with JFreeChart [29] in order to play multimedia files and to draw sensor values at the same time. This way, visualization of video and sensors in synch is possible.

3.3.3 Sharing Component

The generated outputs may be shared over a central server application via uploading / downloading files or sending messages to server. This component has client-server architecture and provides the following features:

- Social Networks integration
- A push-based messaging service
- A web service interface with an advanced upload service

[4] Microsoft DirectShow

3.3.3.1 Social Networks Integration

Twitter and Flickr are currently integrated to the framework as part of social network integration. For example, after an automatic extraction of a video frame, the developed application may easily send this frame as a Twitter image post with a single line of code. The developer first generates a new application on the selected social network and provides the details of the application (api_key, api_secret) to the framework. Using the integrated libraries, the application may then perform actions such as sharing a status, sending a tweet, uploading an image, getting the friends/followers/followed list, getting the status/tweets of a friend/followed, and other actions available through the interfaces provided by social networks.

3.3.3.2 Push Service

Due the mobile and dynamic environment, the use of push mechanism is encouraged, and the framework provides this mechanism as a reusable component. Using this mechanism, the mobile devices may efficiently communicate with the server, using minimum energy and data transfer. On the server side, MQTT server application is started for a predefined port, and the clients are connected to this port when the mobile application is started. The server is also an MQTT client and connects to MQTT server as a regular client.

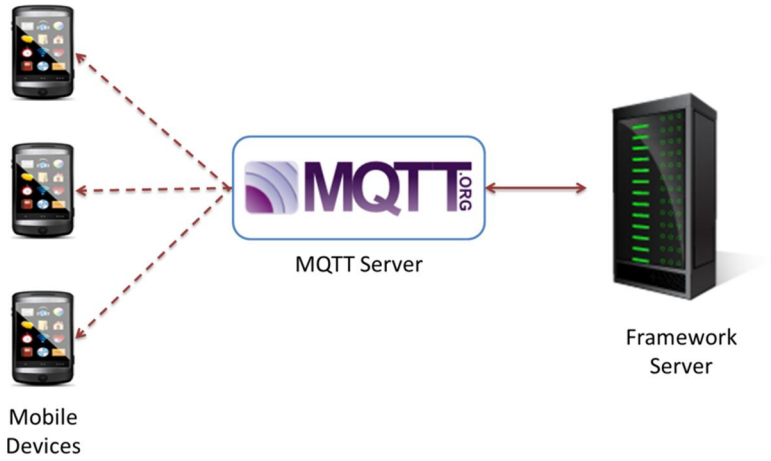


Figure 18 Framework's Push Mechanism

3.3.3.3 Web Services

The web service interface is implemented as a separate web application and is also part of the framework. The server interfaces are developed as “web services” and the REST architecture is used in order to minimize the implementation of web service usage on mobile devices. The Java API for RESTful Web Services (JAX-RS) is used as the reference implementation for JSR 311 specification which defines the REST support. JAX-RS uses annotations to define the REST relevance of Java classes.

This framework’s server side implementation includes web services for user operation (registration, login/logout), traffic services (register location, notify hazard, get hazard

detail, inform all users) and upload services (upload, synchronize, get state of the upload).

CHAPTER 4

PROTOTYPE IMPLEMENTATION AND TESTING

The prototype implementation is a mobile application to be run on road while driving, which aims to automatically detect hazards of the road (potholes, speed bumps and irregular parts of the road), extract the image of the hazard and inform nearby drivers about the hazard. With a collaborative detection and warning mechanism, every driver is warned about incoming hazardous events while they are driving. Technically, the system reads sensor data from the accelerometer, magnetometer and camera sensors of a mobile device mounted on the front windshield. The sensor data is then analyzed and recorded after the analysis. The analysis of sensor data tries to detect hazardous events according to accelerometer sensor value changes during the drive.

Different events in concern are manually examined and the patterns of accelerometer values are identified. After an event is detected, either the image of the event is extracted or the recorded video is split so that the portion which contains the event is extracted. For the latter one, the extracted part's duration is determined according to the speed of the vehicle. For this part of the video, obstacle detection algorithms may be run (which is not implemented as it is not in the scope of this study). After the obstacle is identified (either by accelerometer sensor analysis or image analysis), the event is sent to a central application, which is also developed in this study. The central application analyzes the location of the event and locates nearby drivers that are using the system and sends the event's information to these drivers by a push message. The severity of the event determines which users are to be notified, according to user's own preferences. The mobile application of other drivers receives the push message and warns the driver by visual or audio alerts.

4.1 Road Hazard Detection and Image Extraction Implementation

The main goal of the sample implementation is to develop a multi modal sensor analysis system in order to detect hazardous events of drivers in traffic, and inform nearby drivers about the event by providing as much information as possible. This information would include the exact location and type of event, the severity of the hazard, and (if any) the detected obstacle type and image or video part of the incident encountered. The target users of the sample implementation can be any user driving in traffic. In order to achieve the selected goal for this type of target users, it is necessary that the developed systems should be available with minimum cost and also easily deployable.

The developed system comprises of a mobile application that runs on a device mounted to the vehicle, and a server application that clients connect to. The mobile application collects the data from sensors and makes analysis for the aim of detecting hazardous events, in real time. These possible hazardous events may be “pothole detection”, “bump detection” or “obstacle detection”. For pothole and bump detection, the characteristics of the event are extracted (i.e. width, height, severity). For obstacle detection, a screenshot of the obstacle is captured and if possible the type of the obstacle is extracted (i.e. box, pedestrian, vehicle, etc…). Once the event is detected, the video that is being collected is also processed, and a reference image is extracted from the video.

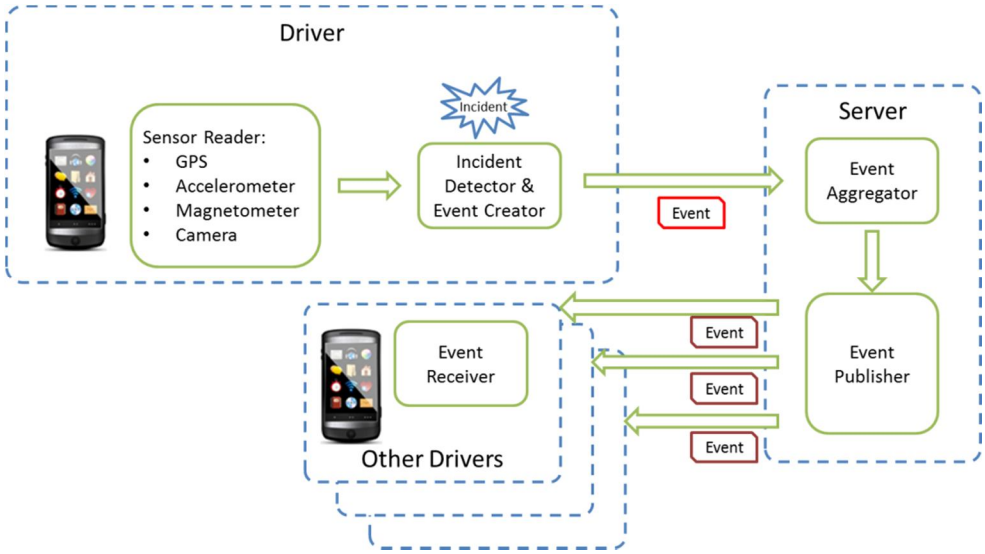


Figure 19 System architecture of sample application

The results are submitted to server application just after the detection is made. The server application receives events from multiple sources and aggregates these events. The aggregation is performed due to possible submission of the same event by different clients. Thus, the events should be analyzed and duplicates of the same event reported by different clients should be identified. After the aggregation step, the server application locates nearby users and informs them instantly about the hazardous incidents. This data includes the location, type, severity and a picture of the event.

4.1.1 Sensor Reading & Device Reorientation

Today, most smart phones include the following sensors:

- GPS: for detecting the location and the speed of the device
- Accelerometer: for detection of hazardous incident
- Magnetometer: for detection of the direction of the device
- Camera: for recording the event and extract the possible cause of the incident

In order to identify the hazardous events (pothole, bump, obstacle), the sensors of the mobile device should be tracked in near time. The primary sensor to be monitored will be the accelerometer because the study is focused on detecting different events based on acceleration changes in 3 axes. The sensor sampling frequency should be high enough

(order of 50 Hz) in order to make efficient detections. The details of the detection algorithms are mentioned in 4.1.2.

Mounting the mobile device to the vehicle is a very important step in order to obtain desired sensor values. The device should be fixed to the vehicle and reflect the exact movement of the vehicle. For this reason, it is a good idea to use a dock station or mobile device holder that is attached to the front window of the vehicle and fasten tightly the mobile device to the holder. This operation is necessary otherwise rattling mobile device's accelerometer sensor may produce incoherent values generated by the movement of the mobile device itself.

However, fixing the mobile device is not adequate to get exact vehicle motion values since the mobile device's coordinate system would probably not aligned with the vehicle's coordinate system. For this reason, it is necessary to re-orient the mobile device coordinate system. We can see that some of the previous works covered in 2.1.3 do not consider the re-orientation of the coordinate and assume that the mobile device is mounted in a fixed direction and orientation (usually perpendicular to the road) which requires a special setup, while other systems in previous works applied the reorientation in order to eliminate the restriction of fixed direction and orientation. The reorientation operation is performed using Euler's angle and the calculations are performed according to [10].

For all incident detection algorithms, reoriented accelerometer values are utilized.

4.1.2 Incident Detection

This study focuses on three types of events: pothole, bump and obstacle detection. Manual analysis of accelerometer values shows that each kind of event has its own unique characteristic in terms of changing accelerometer values. Thus different analysis of accelerometer values is conducted for each of the event types.

4.1.2.1 Regular Drive on Smooth Road

Accelerometer values for a drive on smooth road shows a regular fluctuation on all three axes. In Figure 20, Z values vary approximately between 8 and 10, but we do not observe rigid peaks. These variations are small vibrations of the vehicle, which occurs even on a smooth road.

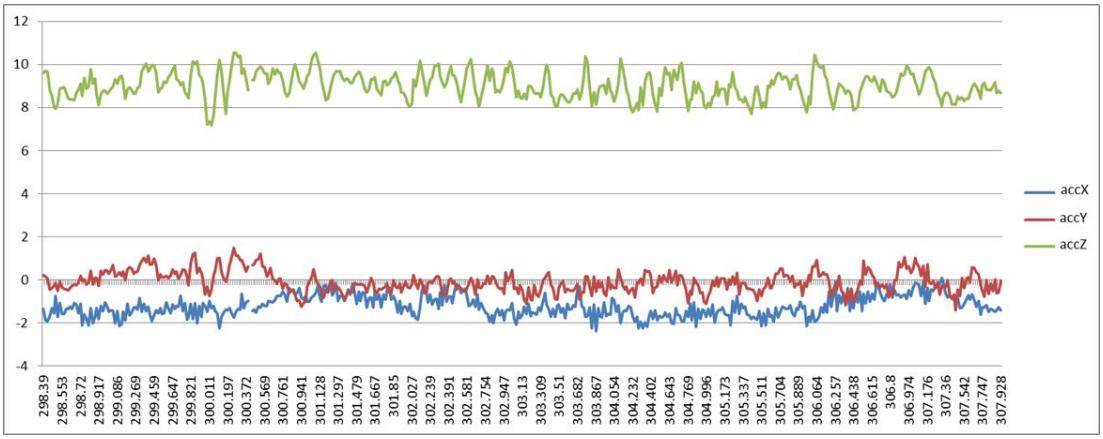


Figure 20 Smooth road raw accelerometer values

4.1.2.2 Severe Deceleration

Severe Deceleration detection algorithm considers the speed decrease with relative to time so that deceleration effect is easily identified. This algorithm is quite simple and tracks the speed changes over time, and applies a high-pass filter to identify abrupt and severe breaks. This detection does not produce a warning by itself, but used within other detection algorithms.

4.1.2.3 Pothole Detection

For the pothole detection, the Z value (the axis perpendicular to the road) of accelerometer sensor is analyzed. Figure 21 shows a typical pothole effect on sensor values.

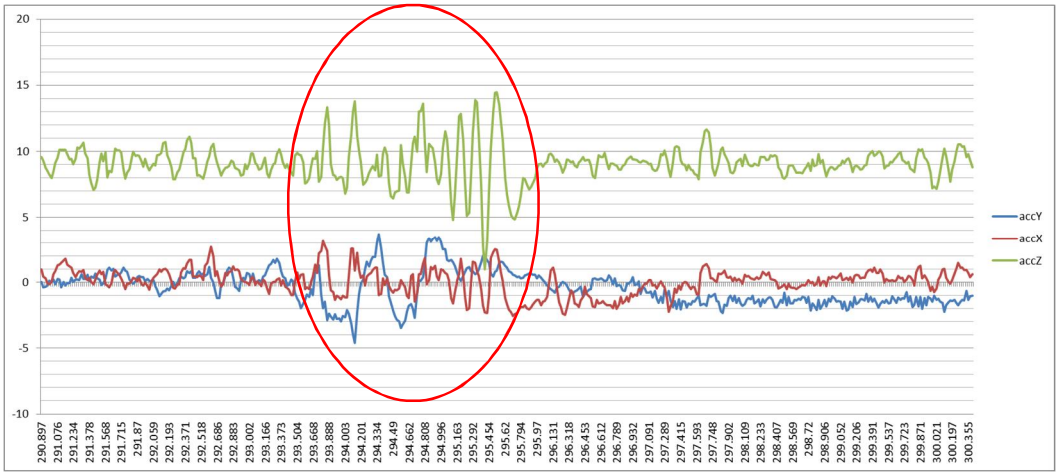


Figure 21 Accelerometer values for a typical pothole

The pothole detection algorithm consists of several operations that run sequentially, and starts with calculating the reoriented accelerometer values.

1. Reorientation: Compute accelerometer values according to reorientation.
2. Velocity Filter: The first filter applied to reoriented values is the simple velocity threshold. The accelerometer values change dramatically over speed of 60 km/h.

They are also affected much at speeds lower than 10 km/h and generate erratic measurements. This study considers only the accelerometer values collected between 10 km/h and 60 km/h.

3. Sensor Values History Length: The vehicle's speed determines the time to pass a pothole. Thus, it passes a pothole faster in high speeds, than in low speeds. Accordingly, the analysis of a pothole considers the speed of the vehicle and a time window is determined according to this speed. For the calculation of the time windows, the equation (1) is utilized on filtered velocity values. C is a coefficient that is defined based on the vehicle characteristics and Wmax is the maximum upper bound of the window. Wmax is a pre-calculated value found as "36" (as a result of the calculation of the minimum speed of 10 km/h of a vehicle passing an average of 2m long speed bump with accelerometer sensor collection frequency at 40 Hz).

$$W = W_{max} - C.\sqrt{v - v_{min}} \quad (1)$$

4. Thresholds Value: There are three different thresholds calculated, for big potholes (T_b), medium potholes (T_m) and small potholes (T_s). The thresholds used in comparing different z values depend also on the speed of the vehicle. High speeds produce much variation in accelerometer values than low speeds. Due to this reason, the threshold is determined according to the speed of the vehicle.
5. Max Diff: The last measurements' z value is compared with other z values in the history where the length of the history is determined in the 3rd step and the maximum difference is found. If the difference absolute value is greater than the thresholds found in step 4th, then this location is marked as a candidate pothole. The detection is performed according to equation (2).

$$Pothole = \begin{cases} Big & if \;\; T_b < Z_{diff} \\ Medium & if \;\; T_m < Z_{diff} <= T_b \\ Small & if \;\; T_s < Z_{diff} <= T_m \\ NoPothole & else \end{cases} \quad (2)$$

6. Bump filter: The candidate potholes hangs for a period of time, which duration is determined again with the speed of the car. If no previous pothole is detected or if no bump detection occurs within this time period, the candidate pothole is marked as a pothole.

4.1.2.4 Bump Detection

The speed bumps present a different sensor value pattern than potholes. A smooth cross of a speed bump generates clearly two peaks and two nadirs (lowest point). The reason for this is quite obvious: first the front tires meet the bump, which produces an initial increase in the height of the vehicle, then follows a decrease when it passes completely. When the front tires complete their course of action, the rear tires produce mostly similar behavior in accelerometer values after a period of time. The process is depicted in Figure 22.

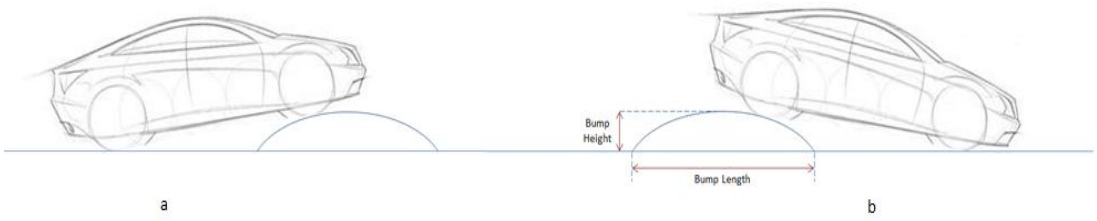


Figure 22 (a) Front tires hit the speed bump (b) Rear tires leave the speed bump

The length and the height of the speed bump are also important and should be considered in the detection of bumps. Although different shaped and sized bumps exhibit some kind of similarity, no single pattern matches all types of bump.

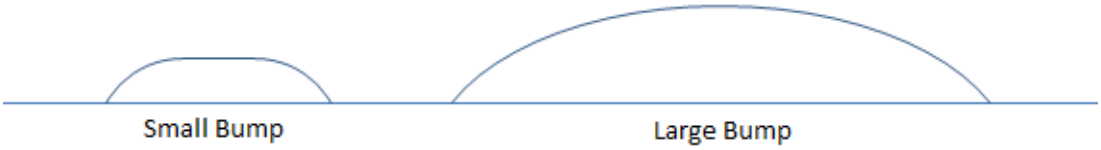


Figure 23 Small and Large Bump Shapes

For small bumps, the timeframe between one local max and one local min value of accelerometer sensor value (front tires pass duration) is small, whereas the other peak and local peaks are generated after a relatively larger time frame (see Figure 24). On the other hand, large speed bumps local max and one local min are much more disjoint and second max comes after a significant period of time of the first local min (see Figure 25).

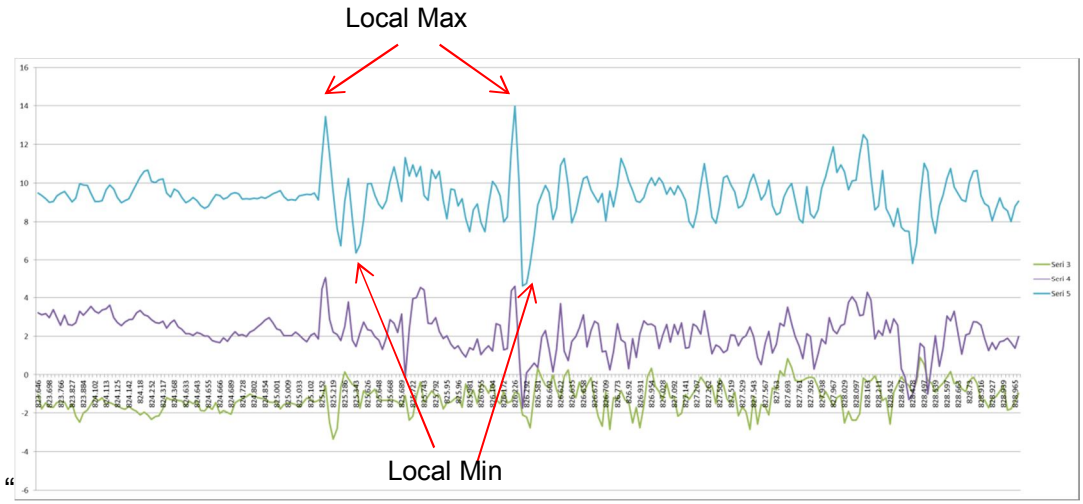


Figure 24 Typical Small Speed bump

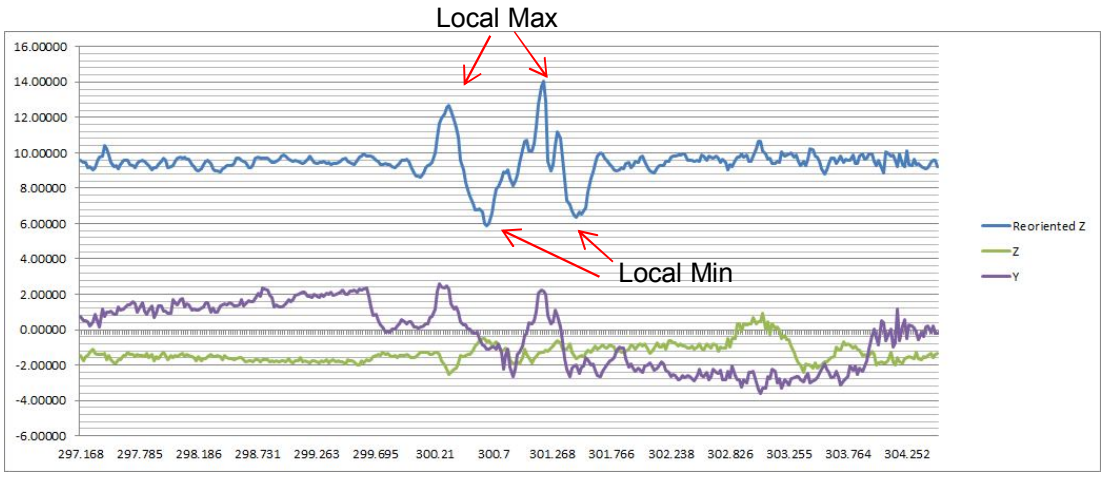


Figure 25 A typical large speed bump

For both kind of bumps (small and large), a sinusoidal pattern is generated. However, small bump's values are disjoint for the front tires' hit and rear tires' hit whereas they are much more adjacent for large bumps. In fact, the period of time between first and second hit of tires determines the length of the bump. The developed algorithm considers different kinds of bumps and tries to extract the bump length information from this fact.

Since the speed bump passing produces a sinusoidal behavior in accelerometer Z values, one of the simplest methods to detect a speed bump is to apply high and low threshold filters sequentially and follow the pattern in 4 steps. As depicted in Figure 26, exceeding a maximum threshold (z-max1), then the minimum threshold (z-min1), then a second maximum threshold (z-max2), and lastly another minimum threshold (z-min2) indicates a crossing of the vehicle over a speed bump.

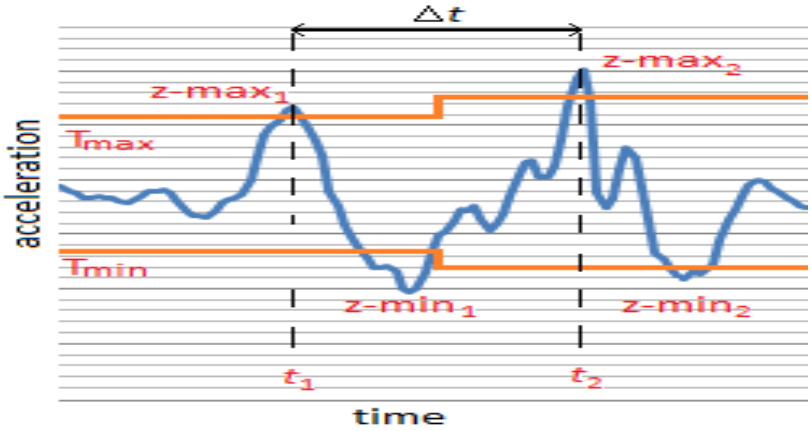


Figure 26 Minimum duration calculation in speed bump detection

According to this general fact, the bump detection algorithm consists of following operations:

1. Reorientation: Compute accelerometer values according to reorientation.
2. Speed Filter: The bump detection algorithm considers also the accelerometer values collected between 10 km/h and 60 km/h, with the same reason as pothole detection mention in 4.1.2.2. A second reason for speed filter in bump detection is that it is unlikely that a driver passes intentionally a bump with a speed higher

than 60 km/h. One of our aims in this study is to inform the user before it hits a bump with high speeds.

3. Threshold Values: High and low thresholds are determined according to the speed of the vehicle. Since the accelerometer values are generated exponentially proportional to the speed, both the maximum and minimum threshold values are calculated accordingly. The thresholds z-max and z-min are calculated according to equations (3), (4), and (5). First, z- avg is calculated as the average of z values within the time window. Then a deviation value is computed according to the velocity (v) of the vehicle multiplied by a coefficient, defined according to the vehicle properties. Then z-max and z-min are calculated according to equations (4) and (5) respectively.

$$d = v * C \tag{3}$$

$$z_{max} = z_{avg} + d \tag{4}$$

$$z_{min} = z_{avg} - d \tag{5}$$

4. Pattern Matching & Time Filter: The pattern is a consecutive threshold exceeding of z-max and z-min twice. However, this pattern may also be generated by a big pothole. The interval (Δt) between z-max1 and z-max2 is very important for the detection of a speed bump, since a small interval value is an indication of other kinds of irregularities (such as pothole, or any other sudden change of road surface), and a too large interval value is a probable indication of two consecutive potholes. Thus, the time interval (Δt) is compared against the threshold value T, and the bump is detected according to the equation (6).

$$Bump = \begin{cases} 1 & if \quad \frac{1}{2}Th < \Delta t < 2Th, \\ 0 & else \end{cases} \tag{6}$$

4.1.2.5 Image Extraction

A distinctive feature of this study is the implementation of the multimodal analysis approach. The study enables the analysis of sensor values in one modality based on the results of sensor values of a different modality. With this approach, the video, which is recorded simultaneously while detecting the hazards, is analyzed and the “important” parts of the video, which contain the hazard scenes, are extracted by using the detection results of the analysis on the accelerometer sensor values.

Since image processing is a time and resource consuming operation, instead of analyzing the whole video from start till the end, only the video parts which have to be focused and which are likely to contain a hazard, are extracted by the multimodal analysis. The video parts and frames are extracted with this method and prepared for

further processing (image processing on extracted frames is not conducted as part of this study).

For the video and frame extraction, the following method is applied: for a detected road hazard, the video that has been recorded is analyzed, and a video section starting from 10 meters ahead of the hazard until the hazard, and an image from 5 meters ahead of the hazard is extracted automatically. In order to achieve this, the exact video location which matches the required distances before the hazardous road segment is identified.

$$X_0 = \sum_{i=0}^{n} V_i.\Delta t_i \tag{7}$$

$$\Delta T_d = \sum_{i=0}^{n} \Delta t_i \tag{8}$$

$$t_{shot} = t_{bump} - \Delta T_d \tag{9}$$

For image extraction, we have selected the best distance for a good image before the bump speed as 5 meters. Assuming X_0 = 5m, the number of sensor values (n) that should be considered backwards to span that distance is found in equation (7). After finding the number of sensor values to be read backwards, the total time passed (ΔT_d) in reading n sensor values is calculated in equation (8).

Figure 27 Image of nine different detected speed bumps, extracted automatically

Since the sensor values and the video are synchronized, the correct location of t_{shot} for the image to be extracted from the video is calculated according to equation (9).

4.1.3 Sharing with Other Drivers

Using the Sharing Component of the framework, the application is able to warn the nearby users about a hazard on the road. Hence, upon a detection of a road hazard, the application immediately creates an event which consists of the time, location, hazard type and hazard severity data. This event is sent to a central server (if the device has connection). The server then runs a quick geo-location search to locate the active drivers nearby the event and sends a notification about the event to these drivers, using push mechanism.

Once the event is received by the vehicle, the application checks whether that vehicle motion is headed towards the event or not. There, we simply check whether the direction between the event location and the vehicle motion direction. If the direction is towards the event, the driver is warned about it, when the distance is less than a threshold (again calculated according to the velocity of the vehicle). The warning may be presented by visual or audio alerts, based on the user preferences.

4.1.3.1 Event Creation

After the successful detection of object and best frame selection, a warning would be raised. The warning of incident includes the following information:

- the time
- the location
- the speed
- the direction
- the type of the event:
    - pothole
    - speed bump
    - object detected
        - box
        - pedestrian
        - stopped car
        - unknown object
    - unknown event
- a photo of the scene

This information is sent to the central server using push mechanism. IBM MQTT [16] is integrated into the mobile application, and the messaging is performed using this library. Server application uses also an MQTT implementation as a receiver endpoint.

4.1.3.2 Event Aggregation & Publishing

The server gets the event, records it and finds other drivers that are geographically nearby the event location. The same information is then sent to these drivers.

The push mechanism requires a receiver on server side, in order to get the messages and publish to subscribers of the same channel. Mosquitto MQTT Message Broker [33] is used to receive push messages and forward to online users nearby. The message broker's source code is updated for the implementation of two new features:

- Event aggregation
- Location of nearby users

In its original form, the message broker does not process the messages and forwards directly to its subscribers. However, there are two reasons why the messages should be processed. The first reason is that the incidents detected may be important for only a short amount of time (such as obstacle detected on a road) and this short amount could be one day for example. But for other type of events, such as severe pothole or a dangerous speed bump, it is possible that the event is detected days ago but a driver using the application newly should be warned when approaching to the location of the hazardous road anomaly. Thus the events are stored for a valid period of time, until no new event is detected for the same location.

The second reason to process the message is that multiple users may alert the same location and the same incident, which may cause duplicates. Thus the server analyze the incident by location and type and it does not forward duplicate incidents to nearby drivers. The location information obtained from GPS devices is not accurate enough to get the exact location for each identical incident. Thus an approximation is performed based on location and direction values (obtained from magnetometer value) of the notified incident.

The notifications that are sent to clients are also stored, in order not to publish the same notification over and over to the same driver. The number of re-notification is configurable on client side, and the server tracks this settings.

4.1.3.3 Event Receiving

Once an event is received by the server application, the message is forwarded to nearby drivers and the mobile application warns the user about the incoming event with visual and audio notifications, which are customizable by the user.

4.2 Mobile Application User Interfaces

4.2.1 "Yol Asist" Application

The developed application name is given as "Yol Asist" (Turkish of "Road Assist") and the installed application is shown on the application list.

Figure 28 Application installed on an Android device

4.2.2 First Usage

If the application is newly installed, before anything else, the first usage screen appears, and requires a name to be entered. The name is used as the identifier of the driver.

Figure 29 Name input screen

After the name successfully entered, a welcome screen appears. This screen also gives some advices on the usage of the application.

Figure 30 Welcome Screen

The terms of use is also included in this text message, and the user should accept the terms of use in order to be able to start using the application.

Figure 31 Warnings and terms of use

4.2.3 Main Screen

The main screen is rather simple. The first button is for viewing already stored records. The second raw contains the following buttons in order: Make a recording, make a recording without camera, make a test recording.

The last raw does include only buttons for testing purposes.

Figure 32 Main Screen

4.2.4 Recording

The applications main functionality is to make recordings and analysis. The screen components are explained in detail in Figure 33. The application executes by pressing “Start” button, and makes the recordings of sensor and video at the same time. The sensor values are analyzed in real-time, and the detections create visual alerts on the screen. The user may see many information about the current status of the execution: the location, the speed, accelerometer values as well as the record duration and number of detected potholes and speed bumps.

The warning level is adjustable, and the user may increase or decrease the level of warning by just setting up the slide bar. The brightness is toggled automatically to save energy, but the user may change this setting.

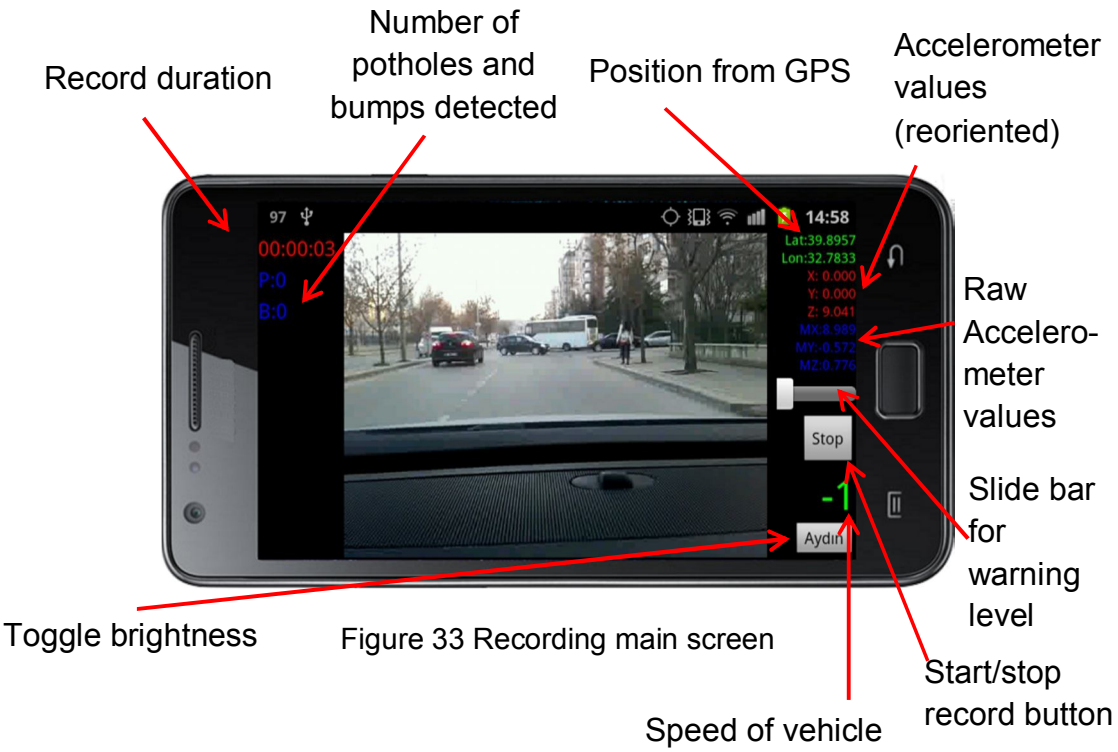


Figure 33 Recording main screen

When the recording is finished, the user presses the “Stop” button, and the system asks for which records to save. Possible different files to be saved are: video file, video chunks created for each pothole or bump detected, an image for each pothole or bump detected, the sensor raw and calculated values and the route file in html showing the locations passed and the pothole / bump detected.

Figure 34 Record save screen

The user may switch the camera to front or rear camera if required, or take a snapshot photo manually, by using the context menu of the record screen.

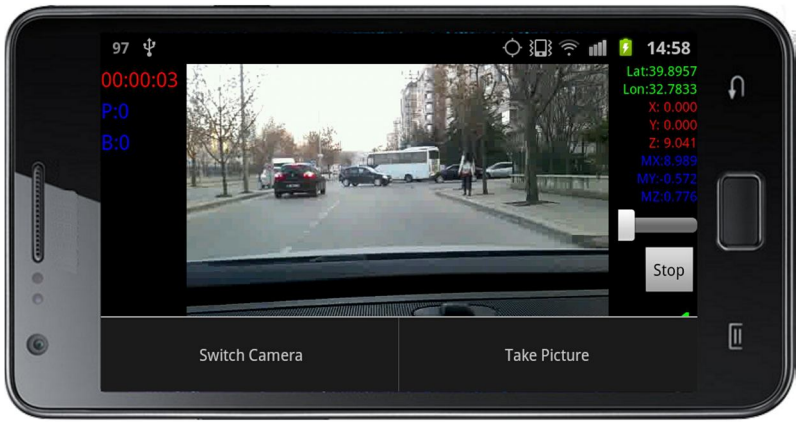


Figure 35 Context Menu of Record Screen

4.2.5 Reviewing Past Records

The stored sensor values, as well as records and other files created during an execution of the application may be reviewed. This functionality first lists each execution as a separate row, and gives brief information about its content.

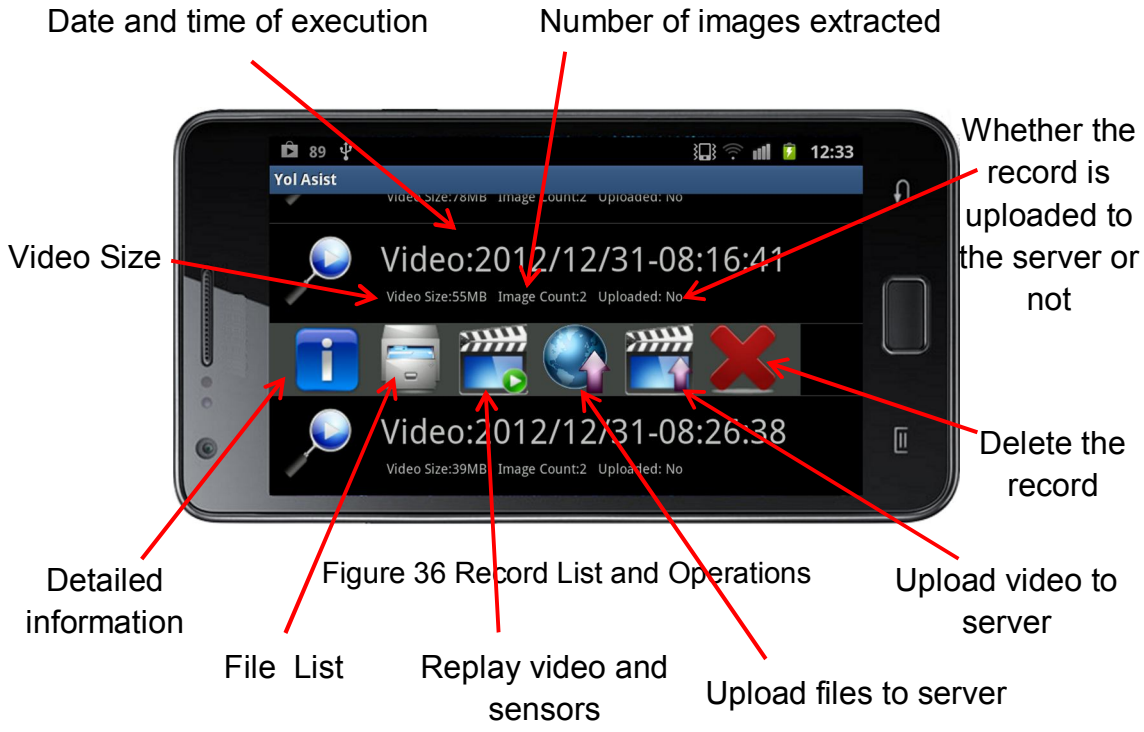


Figure 36 Record List and Operations

The file list button lists all the files that are recorded for the selected execution.

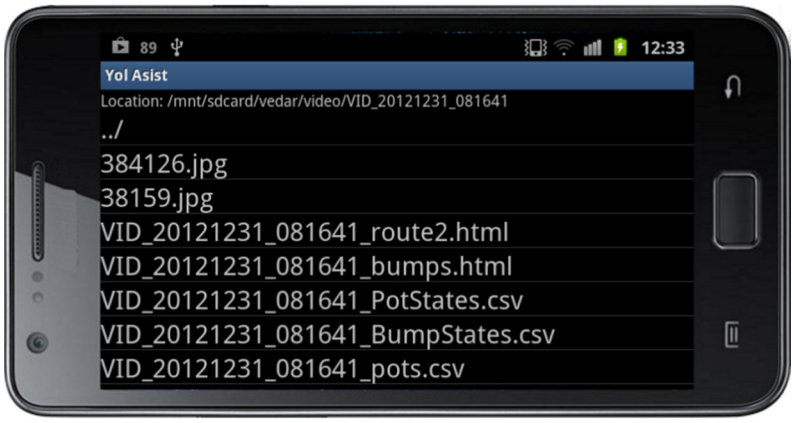


Figure 37 File List of a record

The file list opens up any selected file, with the system level associated program. When the user selects a picture for example, the image viewer opens up the picture. Videos and html files are also rendered with default programs. The sensor values are saved as comma separated values (csv), and viewed in text editor.

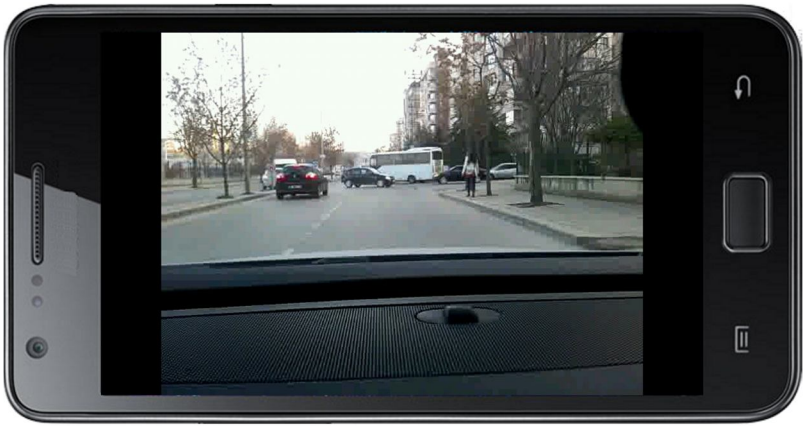


Figure 38 Picture Viewing

If the user presses the “Upload File” button, all the files (except the video file) are uploaded to the server.

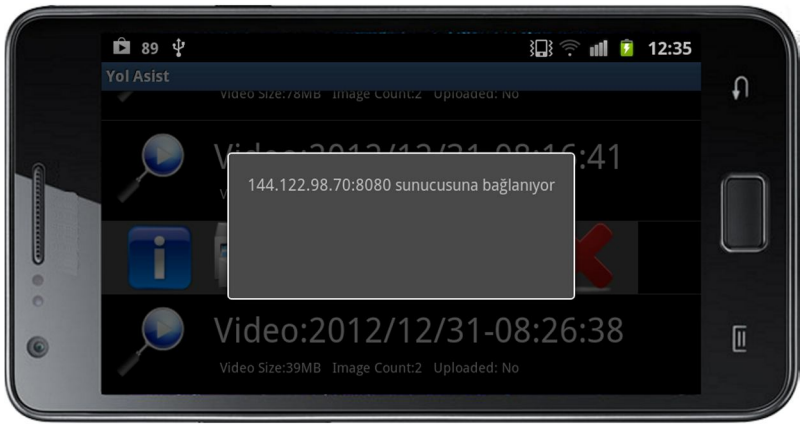


Figure 39 “Upload File” command is first checking the connection

The upload command first checks the connection to the server. The server is configurable in the application.

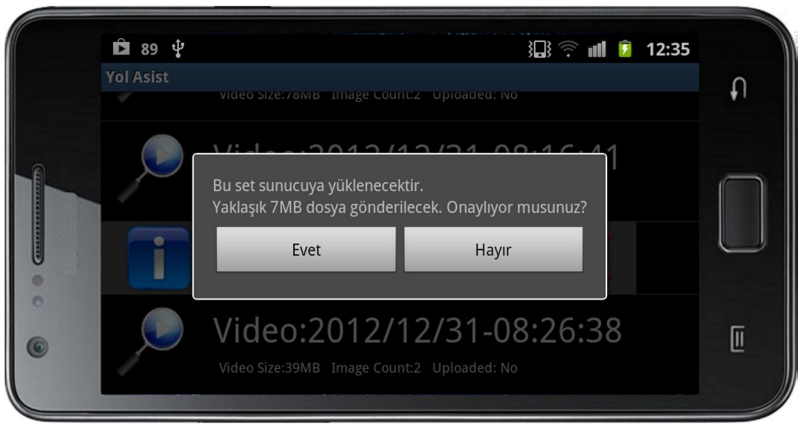


Figure 40 The application asks for permission of transferring data

Then the application calculates the total size of files to be transferred, and asks for permission to send the files. If the user grants permission, then the transfer begins, and the progress bar shows the transfer state.

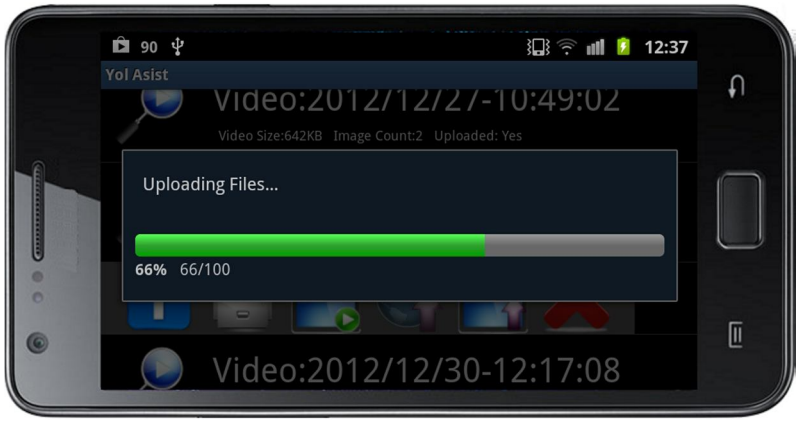


Figure 41 Progress bar of files transfer

The video is transferred using the same procedure, but is activated with a separate button, since video file size is much bigger than other files. It is a good idea to send it separately.

The recorded video as well as sensor values may be replayed. This operation synchronizes the video position with the sensor readings and draws the sensor values using a line chart on the left, while displaying the video on the right. There is also a fast forward button, that locates the incident in the sensor values, and forwards both the video and sensor chart to this position. The user may switch the view to full video, full chart or both, side by side (as seen on the Figure 42).

Figure 42 Replay mode: video and sensor values positions are synch

4.3 Server Application Interfaces

4.3.1 Record Navigation

The server web application allows the user to view the uploaded records via regular web browsers. The users connect to their account and list all their uploaded recordings.

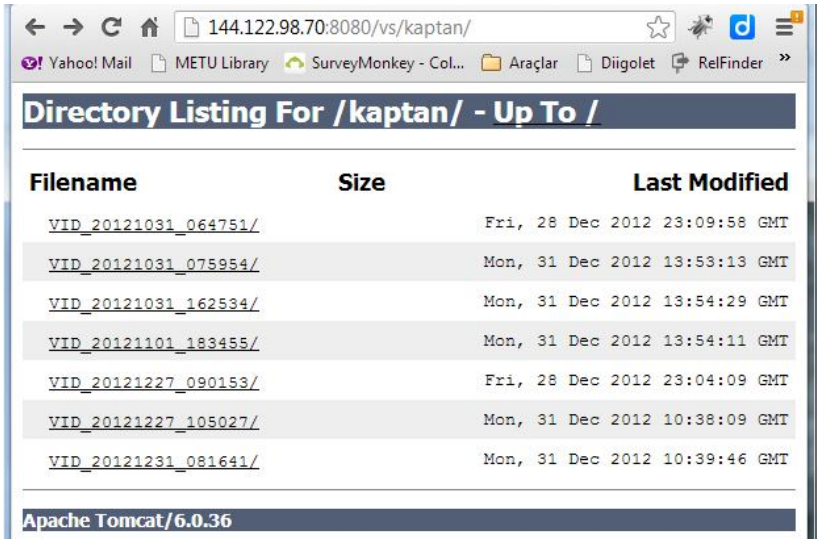


Figure 43 User Kaptan's recordings

Selecting a directory, the files in this directory are listed.

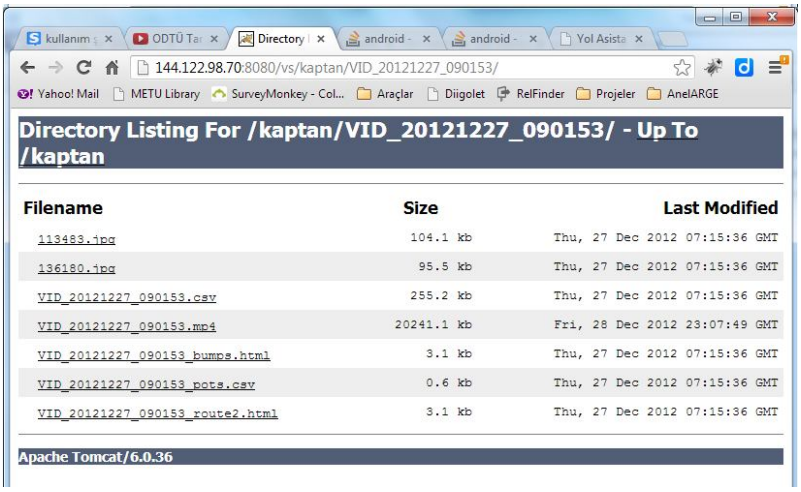


Figure 44 File list of a record

The image files, video file and csv files can be viewed by clicking. The html files created are the route.html and bumps.html. The route.html shows the entire route on a map from start to end, for that execution of the application.

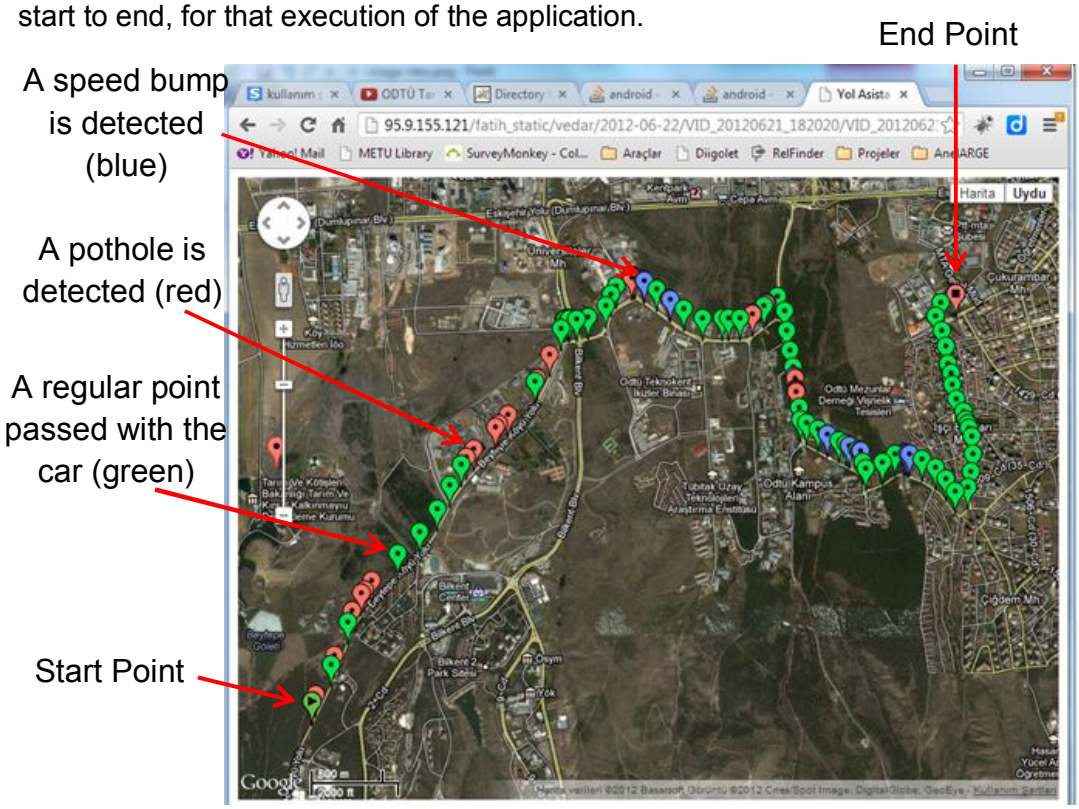


Figure 45 The entire route of an execution can be viewed easily

Once focused on a specific location, the thumbnail image of that event is popped up.

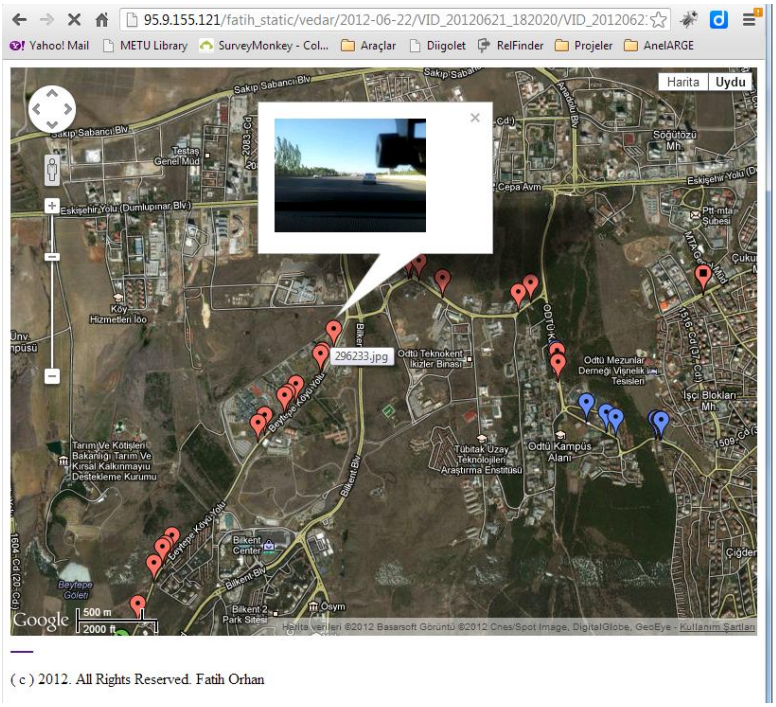


Figure 46 Image Preview

And clicking the image shows it detail.

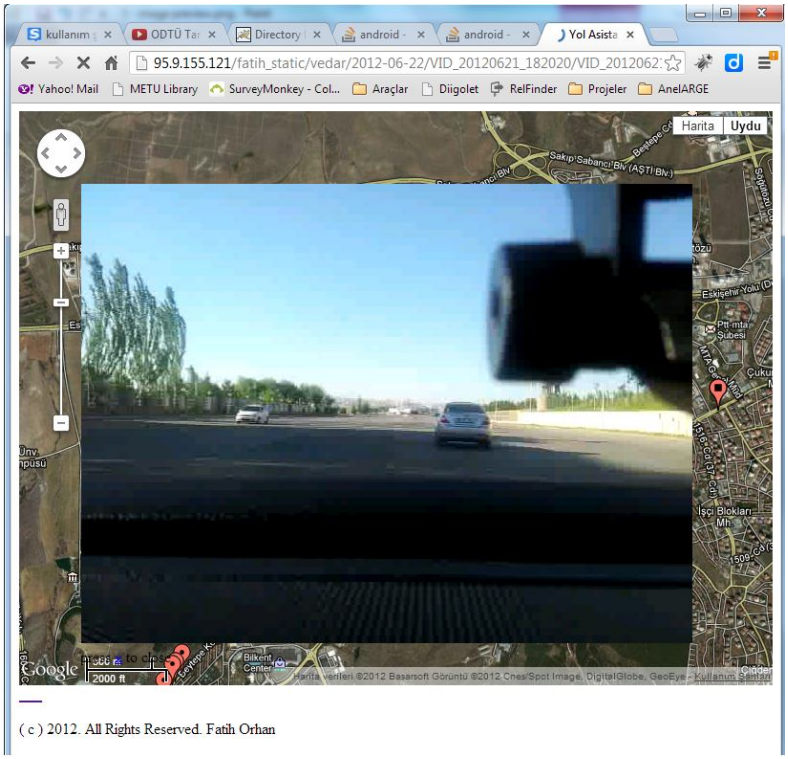


Figure 47 Image Detail

4.3.2 Manual Analysis

For manual analysis of sensor and video recordings, DirectShow Java Wrapper (DSJ) [28], a java abstraction layer around Directshow API5, is modified and integrated with JFreeChart [29] in order to play multimedia files and to draw sensor values at the same time. This way, visualization of video and sensors in synch is possible.

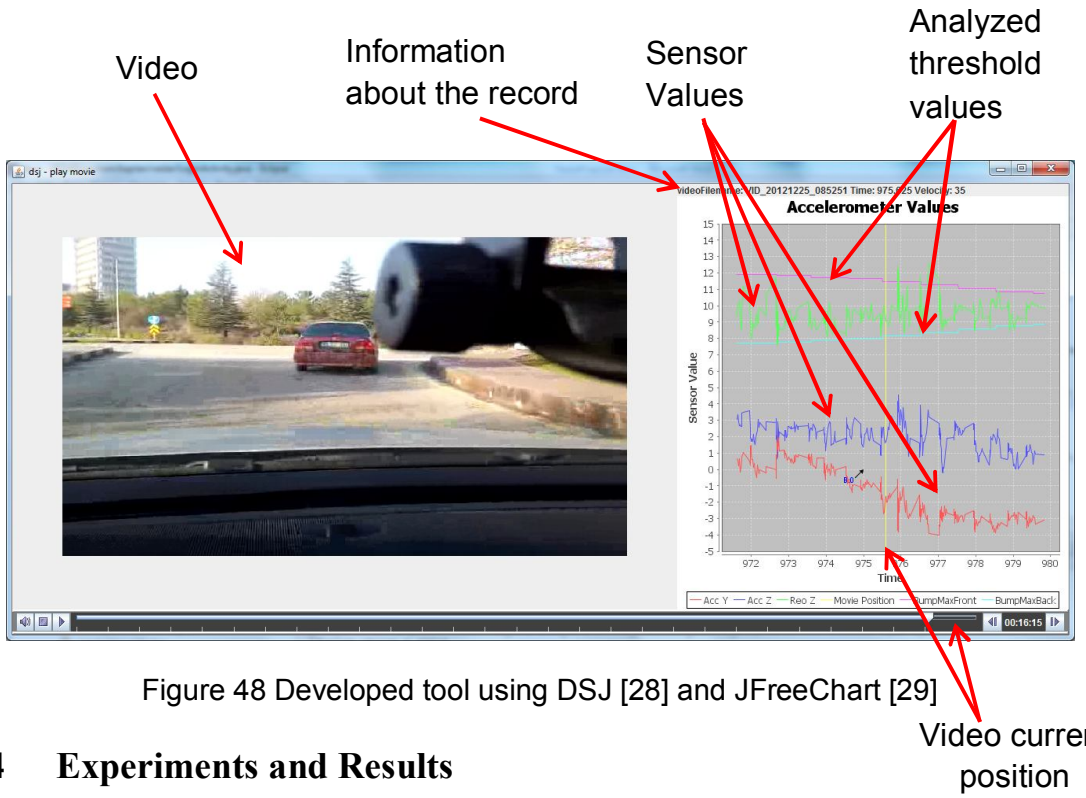


Figure 48 Developed tool using DSJ [28] and JFreeChart [29]

4.4 Experiments and Results

In order to evaluate the performance of the mobile application, a route in our university campus was selected and several driving tests were conducted on this route. In order to build the ground truth for potholes and speed bumps, the correct locations were marked manually before testing, by travelling the entire route. The results were collected from the test runs and compared with the ground truth values. The results were evaluated in terms of recall and precision values.

4.4.1 Mounting the Device

The direction and the orientation of the device should be mounted correctly in order the camera is pointed towards the road and has a clear view of the road. The camera lens should be open and not obstructed in order to capture the scene. On the other hand, there is no constraint on the device to be aligned perpendicular to the ground since the system calculates the precise orientation of the device and reorients the device coordinate system. However the mobile device should be tightly fixed to the device dock in order to prevent generating superfluous acceleration values with the mobile device's motion itself. The convenient mounting of the device is shown in Figure 49.

[5] Microsoft DirectShow

Figure 49 Convenient mounting for a correct camera view

4.4.2 Test Site

The selected test site was inside the campus of Middle East Technical University (METU), and the total road segment is about 6,210 meters. Within the selected segment of the road, 17 bumps and 4 small potholes were detected and manually marked on the map in Figure 50.

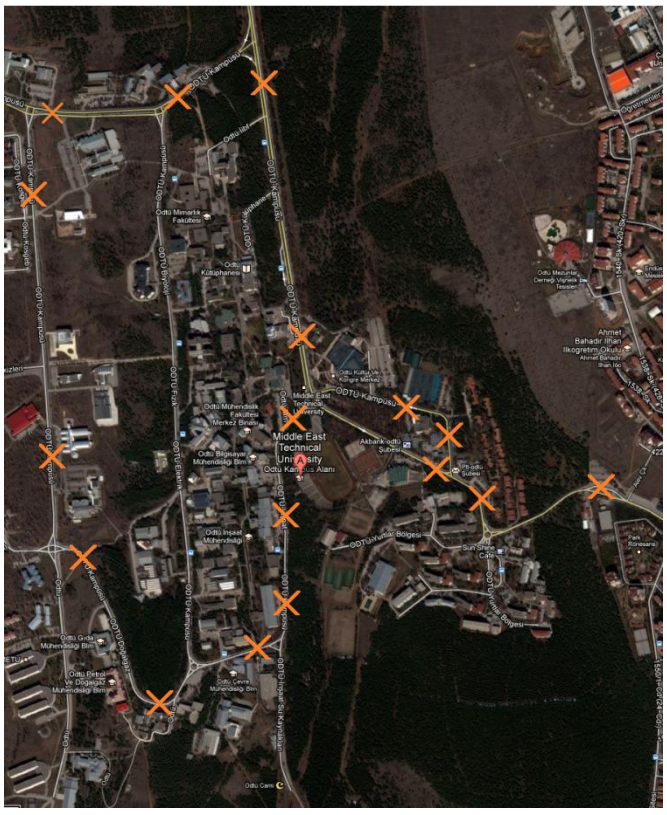


Figure 50 Test Site Map. Potholes and Bumps are manually identified and marked

4.4.3 Collected Data and Analysis

Test drive durations were 25 minutes on average and a total of 25 drives were conducted within the test.

4.5 Results and Discussion

The results of the test drives were collected in comparison with the ground truth values, obtained by manual inspection of the hazards. The performance was measured by precision (true positive detections/all detections) and recall (true positive detections/ground truth).

Table 1 Test Drive Results

Speed Bump Counts and Recall and Precision Results						
Test No	**Ground Truth**	**True Positive**	**False Positive**	**False Negative**	**Recall (%)**	**Precision (%)**
1	11	8	3	3	73	73
2	12	10	3	2	83	77
3	10	9	1	1	90	90
4	11	7	0	4	64	100
5	11	9	2	2	82	82
6	11	8	0	3	73	100
7	9	7	1	2	78	88
8	4	3	0	1	75	100
9	11	11	4	0	100	73
10	11	10	2	1	91	83
11	10	8	1	2	80	89
12	11	9	0	2	82	100
13	11	9	0	2	82	100
14	11	10	0	1	91	100
15	9	7	0	2	78	100
16	10	9	0	1	90	100
17	12	10	0	2	83	100
18	11	10	0	1	91	100
19	11	9	0	2	82	100
20	13	11	0	2	85	100
21	16	13	2	3	81	87
22	10	7	0	3	70	100
23	12	9	0	3	75	100
24	12	11	1	1	92	92
25	11	9	0	2	82	100
AVG	**10.84**	**8.92**	**0.80**	**1.92**	**82**	**93**

The results show an average recall value of 82% and an average precision of 93% (Table I) for successful bump detection. A detailed manual analysis on the results was also conducted, and two speed bumps were identified for which the algorithms mostly failed to detect, thus resulting in false negative. With the manual inspections on these two speed bumps, it has been observed that they are physically different in shape compared to others, such that one side of the speed bump slope was lower than a regular one, thus they produce lesser variations in accelerometer values than expected. For this reason, the algorithms fail to detect the first threshold exceeding, hence fail to detect the speed bump at all. This shows that the algorithms are too sensitive, and needs to be adapted according to different conditions and different shapes of obstacles.

On the other hand, the bandwidth usage for the upload operation was also measured as part of the study. For a typical journey of 25 minutes, the total data to be uploaded is about 8 KB for all the events and 2.2 MB for the extracted pictures (25 hazards, one picture for each, and each picture ~90 KB, with dimensions 640x480). The video fragments extracted are 650 KB on average (~3 seconds) for one event and the total video size is about 8 MB for each minute. From these measurements, it can be seen that hazard events data can be sent at any time, for any event. However, for minimum bandwidth usage, it is better to send the events' extracted images for only those having high severity, and send other images over Wi-Fi after the trip. The video fragments and the total video file are currently only sent over Wi-Fi.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Mobile devices are widely owned and the widely adopted application stores of mobile platforms facilitate the penetration of mobile applications to the mass audience. Many pervasive mobile applications are being utilized in the concept of Intelligent Transportation System and Smart Vehicle and users are becoming more and more connected to each other via social networks and not only consuming but also generating information at the same time. Developing mobile applications for dynamic environments such as on road applications brings many technical and technological complexities. Especially when utilizing the sensors of the mobile devices, the developers struggle with many kinds of problems in order to correctly collect sensor values and perform signal and/or image processing analysis based on these values. Moreover, performing multimodal analysis that utilizes more than one sensor data brings also new complexities. On the other hand, connecting to social networks and getting/sharing information from/to these networks may also pose additional difficulties.

In this study, a multimodal sensor analysis framework is developed to enable vehicular mobile application implementation. The framework enables performing analysis on real-time synchronized sensor values. Utilizing only single modal analysis, regular mobile applications produce only limited results which lack accuracy, consistency or efficiency. Using this framework, the applications easily connect and use the sensor values as well as perform complex analysis using the output of one sensor as an input to another sensor in multimodality. On the other hand, developers usually spend much time to develop regular components which perform basic similar tasks. With this framework, the burden of developing generic components over and over by different developers is discarded, and the developers may concentrate only on the implementation of the analysis they want to perform with sensor values. The sharing component also provides easy and fast access to social networks so that the results of the analysis are easily shared with friends, with nearby users or communities.

Although a sample implementation is performed in this study, the framework is generic and flexible to be used for development of various kinds of sensor-based vehicular mobile applications. Sample applications that may be implemented with this framework include but not limited to: real-time traffic generation and monitoring, automatic photo-blog generator application or journey creation, event data recording, road profiling, automatic sign detection and sharing application or automatic gas price recognition and

sharing application, automatic plate recognition application, anomaly detection (traffic congestion, accident or road up) and sharing with nearby users or authorities and all crowd-sourcing based sensing and sharing applications.

The sample mobile application developed in this study utilizes the framework and detects the potholes and speed bumps on the road, while automatically extracting the image and video section corresponding to the road segment. This extracted section is prepared for further image processing. The detection is shared over a central application with other drivers, which provides a quick awareness about traffic events. With the methods employed in the sample application, the advantages of multimodal analysis over single-modal analysis are exhibited.

The framework development is complete as part of this study, and provides the required functionality as listed. As future work, many different research and development improvements may be performed on that framework as well as optimizations on different topics. First of all, it is planned to increase the flexibility and robustness of the framework by developing new sample applications and extending the interfaces as needed. As part of sensor integration, new internal sensors (light, proximity, gyroscope) as well as external sensor such as heart rate monitoring, temperature, pressure for driver's health or external sensors for connecting to vehicle self-diagnostic and reporting interfaces (such as OBDII). Moreover, the interfaces of the framework may be improved to enable built-in complex analysis and already developed components for faster development. Examples of such components are interfaces for predefined vehicular action based on accelerometer values (vehicle stopped, moving slowly, moving fast, sudden turn, abrupt deceleration etc).

On the other hand, interfaces are not tuned for efficient use of device resources. Especially GPS sensor and camera consume much energy. As part of future work, smart algorithms should be employed in order to reduce the GPS energy consumption and alternative methods may be developed to track the location of the user. On the other hand, camera recording does not necessarily need to display the video content to the driver while driving. Thus background recording may be investigated and developed as part of the energy consumption improvement. Considering other sources of consumption, efficient usage of battery should be analyzed carefully and studies to reduce the energy consumption of the framework should be investigated.

REFERENCES

[1] Profilometer, http://en.wikipedia.org/wiki/Profilometer, Retrieved: 29.11.2012

[2] Sayers, M.W., Karamihas S. M, The Little Book of Profiling, University of Michigan Transportation Research Institute, 1998

[3] International Roughness Index, http://en.wikipedia.org/wiki/International_Roughness_Index, Retrieved: 29.11.2012

[4] Dynatest 6450 Lightweight Profilometer, www.dynatest.com/pdf/6450.pdf, Retrieved: 29.11.2012

[5] International Cybernetics, Lightweight Profiler, http://www.intlcybernetics.com/images/PDFs/2,012-010%20Road%20Profilers%20Brochure.pdf Retrieved: 29.11.2012

[6] Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. In: MobiSys'08. pp. 29-39, 2008

[7] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: using mobile smartphones for rich monitoring of road and traffic conditions," in Proceedings of the 6th ACM conference on Embedded network sensor systems, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 357–358.

[8] Mohan P, Padmanabhan V. N., and Ramjee R, "Trafficsense: Rich monitoring of road and traffic conditions using mobile smartphones," Microsoft Research, Tech. Rep. MSR-TR-2008-59, April 2008.

[9] Mednis A., Strazdins G., Zviedis R., Kanonirs G., Selavo L., Real time pothole detection using android smartphones with accelerometers. Distributed Computing in Sensor Systems and Workshops, International Conference on June, 2011

[10] Astarita V, Caruso M.V., Danieli G., Festa D.C., Giofrè V. P., Iuele T., Vaiana R., A mobile application for road surface quality control: UNIquALroad, Procedia - Social and Behavioral Sciences 54, 2012

[11] Jain M., Singh A. P., Bali S. and Kaul S.. Speed-breaker early warning system. Proceedings of the 6th USENIX/ACM Workshop on Networked Systems for Developing Regions, New York, NY, USA, 2012. ACM.

[12] Tai Y., Cheng-wei C., Jane Y. H., Automatic Road Anomaly Detection Using Smart Mobile Device. In Proceedings of the 15th Conference on Artificial Intelligence and Applications (TAAI 2010). November, 2010.

[13] Chen K., Lu M., Fan X., Wei M. and Wu J., Road Condition Monitoring Using On-board Threeaxis Accelerometer and GPS Sensor, Communications and Networking in China (CHINACOM), 2011

[14] DoD, GPS Navstar, Global Positioning System, Standard Positioning Service (SPS), Performance Standard, p.11, 2008

[15] Google Cloud Messaging for Android, http://developer.android.com/google/gcm/index.html, Retrieved: 22.12.2012

[16] IBM MQ Telemetry Transport, http://mqtt.org/, Retrieved: 22.12.2012

[17] The Deacon Project, http://deacon.daverea.com/, Retrieved: 22.12.2012

[18] OpenCV Homepage, http://opencv.willowgarage.com/, Retrieved: 27.12.2012

[19] FFmpeg Homepage, http://ffmpeg.org/, Retrieved: 27.12.2012

[20] Türkiye İstatistik Kurumu (TUİK), Traffic Accident Statistics 2011, http://www.turkstat.gov.tr/Kitap.do?metod=KitapDetay&KT_ID=15&KITAP_ID=70 p. 1, 2012

[21] A. Fenner, "Placing value on Information", Library Philosophy and Practice, Vol. 4 No. 2, Spring 2002

[22] State-Based Costs of Deaths from Crashes, http://www.cdc.gov/Motorvehiclesafety/statecosts/index.html/, Retrieved: 29.12.2012

[23] Motor Vehicle Accidents—Number and Deaths: 1990 to 2009, U.S. Census Bureau, Statistical Abstract of the United States: 2012 http://www.census.gov/compendia/statab/2012/tables/12s1103.pdf, Retrieved: 29.12.2012

[24] Küçükay, F., Bergholz, J.: Driver Assistant Systems. In: International Conference on Automotive Technologies, Turkey (2004)

[25] The Economic Impact of Motor Vehicle Crashes 2000, National Highway Traffic Safety Administation, U.S Department of Transportation, http://www.cita-vehicleinspection.org/Portals/cita/autofore_study/LinkedDocuments/literature/NHTSA%20the%20economic%20impact%20of%20motor%20vehicle%20crashes%202000%20USA%202002.pdf , Retrieved: 30.12.2012

[26] iOS and Android Adoption Explodes Internationally, Flurry Report, http://blog.flurry.com/bid/88867/iOS-and-Android-Adoption-Explodes-Internationally , Retrieved: 30.12.2012

[27] A Chart Engine Homepage, http://www.achartengine.org/index.html, Retrieved: 30.12.2012

[28] DSJ - DirectShow <> Java wrapper homepage, http://www.humatic.de/htools/dsj.htm, Retrieved: 30.12.2012

[29] JFreeChart Homepage http://www.jfree.org/jfreechart/, Retrieved: 30.12.2012

[30] Funf homepage, http://funf.org/, Retrieved: 31.12.2012

[31] N. Aharony, W. Pan, C. Ip, I. Khayal, A. Pentland, Social fMRI: Investigating and shaping social mechanisms in the real world, Pervasive and Mobile Computing, 2011, ISSN 1574-1192, 10.1016/j.pmcj.2011.09.004.

[32] Open Data Kit Homepage, http://opendatakit.org/, Retrieved: 31.12.2012

[33] Mosquitto Homepage, http://mosquitto.org/, Retrieved: 31.12.2012

[34] Sha, Wenjie, et al. "Social Vehicle Navigation: Integrating Shared Driving Experience into Vehicle Navigation." (2013).

[35] Android Mobile Context Instrumentation Framework, AWARE Homeage, http://www.awareframework.com/home/, June. 27, 2013, Retrieved July 06, 2013

[36] DailyRoads Voyager Homepage http://www.dailyroads.com/voyager.php, July 06. 27, 2013, Retrieved July 06, 2013

[37] Safe Start Google Playstore Application homepage, https://play.google.com/store/apps/details?id=net.waterstart.waterboxfree#?t=W251bGwsMSwxLDIxMiwibmV0LndhdGVyc3RhcnQud2F0ZXJib3hmcmVlIl0. July 06. 27, 2013, Retrieved July 06, 2013

[38] Goecke, R., Pettersson, N., & Petersson, L. (2007, June). Towards detection and tracking of on-road objects. In Intelligent Vehicles Symposium, 2007 IEEE (pp. 416-421). IEEE.

[39] Koch, C., & Brilakis, I. (2011). Pothole detection in asphalt pavement images. Advanced Engineering Informatics, 25(3), 507-515. Wijnhoven, R. G. (2011, November).

[40] Beucher, S., & Bilodeau, M. (1994, October). Road segmentation and obstacle detection by a fast watershed transformation. In Intelligent Vehicles' 94 Symposium, Proceedings of the (pp. 296-301). IEEE.

[41] J. Langheim. Sensing of car environment at low speed driving. In 7th World Congress on Intelligent Transport Systems, Torino, , Oct. 2000.

[42] Lefaix, G., Marchand, T., & Bouthemy, P. (2002). Motion-based obstacle detection and tracking for car driving assistance. In Pattern Recognition, 2002. Proceedings. 16th International Conference on (Vol. 4, pp. 74-77). IEEE.

[43] It's Official: Google Acquires Crowdsourced Navigation App 'Waze', http://www.wired.com/gadgetlab/2013/06/google-waze-acquisition/, 11.06.2013, Retrieved: 11.07.2013

APPENDIX

Multimodal Sensor Framework Interface Definitions

1. Sensing Component

a. ISensorListener interface

This interface is defined in order to deliver the collected sensor values to the developer implementation. The values may be passed as raw values or computed/aggregated values.

```
public interface ISensorListener {

	public static final int SENSOR_READING_TYPE_RAW = 1;
	public static final int SENSOR_READING_TYPE_AGGREGATED = 2;
	public static final int SENSOR_READING_TYPE_SAMPLED = 4;
	public static final int SENSOR_READING_TYPE_MEAN = 8;
	public static final int SENSOR_READING_TYPE_MEDIAN = 16;
	public static final int SENSOR_READING_TYPE_REORIENTED = 32;

	public int getSensorReadingType();
	public int getSensorWindowSize();

	public void updateAccelerometer(long measureTime, Double accX,
	Double accY, Double accZ, Double reorientedX, Double reorientedY,
	Double reorientedZ);

	public void updateMagnetometer(long measureTime, Double magX,
	Double magY, Double magZ);

	public void updateGPSSensors(long measureTime, Double latitude,
	Double longitude, Double velocity);

	public void updateVelocity(long measureTime, Double velocity);
}
```

b. ICameraPreviewListener interface

This interface is defined in order to provide the frames capture by the camera to the developer implementation. It also provides the video file location in case of video recording.

```
public interface ICameraPreviewListener {
```

```
	public static final int PREVIEW_FRAME_TYPE_VIDEO = 1;
	public static final int PREVIEW_FRAME_TYPE_BYTEARRAY = 2;
	public static final int PREVIEW_FRAME_TYPE_BITMAP = 3;
	public static final int PREVIEW_FRAME_TYPE_BITMAPFILE = 8;

	public void initialized(long startTime);
	public int getPreviewFrameType();
	public void onPreviewFrame(long measureTime, byte[] data);
	public void onPreviewFrame(long measureTime, Bitmap bitmap);
	public void onPreviewFrame(long measureTime, String
	bitmapFilepath);
	public void onVideoRecordCompleted(String videoFilepath);
}
```

2. Analysis Component

a. ASensorAnalyzer Abstract Class

This abstract class defines and partially implements an analyzer for the analysis component. The developer may update all public methods of the abstract class or use it as it is.

```
/**
 * This is the abstract class in Analysing Component that provides
 * interfaces for analyzers. It also implements ISensorListener for
 * integrating with Sensing Component. It includes a list of
 * ISensorAnalyzerListener for integrating with Sharing Component.
 * @author fatih.orhan
 * 17 Dec 2012 05:38:54
 */
public abstract class ASensorAnalyzer implements ISensorListener {

	//List of Analyzer Listeners
	private List<ISensorAnalyzerListener> analyzerListeners;
	//Whether the analyzer is in Run or Test mode
	private boolean testMode;
	//Whether the analyzer is currently running or not
	private boolean running;
	//The first sensor logging time
	private long firstLoggingTime;

	/**
	 * Default constructor
	 */
	public ASensorAnalyzer() {
		this.testMode = false;
		this.running = false;
		this.analyzerListeners = new
		ArrayList<ISensorAnalyzerListener>();
	}
	/**
	 * Start the analyzer
	 * @param sensorAnalyzeStartTime
	 */
	public void startAnalyzer(long sensorAnalyzeStartTime) {
```

```
            this.firstLoggingTime = sensorAnalyzeStartTime;
            for(ISensorAnalyzerListener listener:
            getAnalyzerListeners()) {
                  listener.init(firstLoggingTime);
            }
            this.running = true;
      }
      /**
       * Stop the analyzer
       */
      public void stopAnalyzer() {
            setRunning(false);
            for(ISensorAnalyzerListener listener:
            getAnalyzerListeners()) {
                  listener.finish();
            }
      }


      /**
       * Getters & Setters
       */
      public void addAnalyzerListener(ISensorAnalyzerListener listener)
      {
            analyzerListeners.add(listener);
      }

      public List<ISensorAnalyzerListener> getAnalyzerListeners() {
            return analyzerListeners;
      }

      public void setTestMode(boolean testMode) { this.testMode =
      testMode; }

      public boolean isTestMode() {return testMode;}

      public void setRunning(boolean running) {this.running = running;}

      public boolean isRunning() { return running;}

      //ISensorListener interfaces
      public void updateAccelerometer(long measureTime, Double accX,
      Double accY, Double accZ, Double reorientedX, Double reorientedY,
      Double reorientedZ) {}

      public void updateMagnetometer(long measureTime, Double magX,
      Double magY, Double magZ) {}

      public void updateGPSSensors(long measureTime, Double latitude,
      Double longitude, Double velocity){}

      public void updateVelocity(long measureTime, Double velocity) {}

}
```

3. Sharing Component

This component defines several web services which are grouped under “Login services”, “Traffic Services” and “Upload Services”

a. Login Services

The login services are implemented as follows:

- Register User
- Login
- Logout

The signatures of the methods implemented are as follows:

```
@Path("/login")
public class LoginServices {

	@GET
	@Path("register")
	@Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
	public String register(@QueryParam(JSONTags.USERNAME) String username);

	@GET
	@Path("login")
	@Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
	public String login(@QueryParam(JSONTags.USERNAME) String
	username,@QueryParam(JSONTags.PASSWORD) String password);

	@GET
	@Path("logout")
	@Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
	public String logout(@QueryParam(JSONTags.USERNAME) String username);
}
```

b. Traffic Services

The traffic services are implemented as follows:

- Register Location
- Notify Hazard
- Get Hazard Detail
- Inform All Users

The signatures of the methods implemented are as follows:

```
@Path("/traffic")
public class TrafficServices {
	@GET
	@Path("registerLoc")
	@Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
	public String registerLocation(@QueryParam(JSONTags.USERNAME) String
	username,@QueryParam(JSONTags.LATITUDE) Double
	latitude,@QueryParam(JSONTags.LONGITUDE) Double
	longitude,@QueryParam(JSONTags.TRIPID) Long tripId);

	@GET
	@Path("notifyHazard")
	@Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
	public String notifyHazard(@QueryParam(JSONTags.USERNAME) String
	username,@QueryParam(JSONTags.LATITUDE) Double
	latitude,@QueryParam(JSONTags.LONGITUDE) Double longitude,
	@QueryParam(JSONTags.SEVERITY) Double severity,
	@QueryParam(JSONTags.TYPE) Integer type,@QueryParam(JSONTags.DIRECTION)
	Double direction);

	@GET
	@Path("getHazard")
	@Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
	public String getHazard(@QueryParam(JSONTags.USERNAME) String
	username,@QueryParam(JSONTags.ID) Long
	hazardId,@QueryParam(JSONTags.IDLIST) List<Long> hazardIds);

	@GET
	@Path("informAllUsers")
	@Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
	public String informAllUsers(@QueryParam(JSONTags.USERNAME) String
	username);

}
```

c. Upload Services

The upload services are implemented as follows:

- Upload File
- Synchronize
- Get state of the uploaded file
- Is directory uploaded

The signatures of the methods implemented are as follows:

```
@Path("/file")
public class UploadService {

	@POST
	@Path("upload")
	@Produces("text/plain;")
```

```
        public String uploadFile(@QueryParam(JSONTags.SESSION) String session,
        @QueryParam(JSONTags.USERNAME) String
        username,@QueryParam(JSONTags.FILE_LASTUPDATETIME) String
        lastUpdateTimeString, @Context HttpServletRequest request, InputStream
        fileInputStream);

        @GET
        @Path("synchronize")
        @Produces(MediaType.APPLICATION_JSON)
        public String synchronize(@QueryParam(JSONTags.SESSION) String session,
        @QueryParam(JSONTags.USERNAME) String
        username,@QueryParam(JSONTags.DIRECTORY) String directory);

        @GET
        @Path("getstate")
        @Produces(MediaType.APPLICATION_JSON+";charset=UTF-8")
        public String getState(@QueryParam(JSONTags.SESSION) String session);

        @GET
        @Path("isdiruploaded")
        @Produces(MediaType.APPLICATION_JSON)
        public String isDirectoryUploaded(@QueryParam(JSONTags.SESSION) String
        session, @QueryParam(JSONTags.USERNAME) String
        username,@QueryParam(JSONTags.DIRECTORY) String directory);
}
```

TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü ☐

Sosyal Bilimler Enstitüsü ☐

Uygulamalı Matematik Enstitüsü ☐

Enformatik Enstitüsü ☑

Deniz Bilimleri Enstitüsü ☐

YAZARIN

Soyadı : Orhan…….
Adı : Fatih…….
Bölümü : Enformatik Enstitüsü – Bilişim Sistemleri……...

TEZİN ADI (İngilizce) : A MULTIMODAL SENSOR ANALYSIS FRAMEWORK FOR VEHICULAR MOBILE APPLICATIONS ……
……..…....

TEZİN TÜRÜ : Yüksek Lisans ☑ Doktora ☐

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokop alınabilir. ☐
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
3. Tezimden bir (1) yıl süreyle fotokopi alınamaz. ☑

TEZİN KÜTÜPHANEYE TESLİM TARİHİ: …….