Integral IoT-Based Visible Platform —Smart Vehicles Controlling System

Prepared by:

Huang Luyao

Chen Sicong

Jin Xin

Qiu Hui

Summer School of Machine Learning, Robotics and Sensor Networks

Imperial College London

Date: 13/8/2015

ABSTRACT

The report mainly focuses on the establishment of an integral IoT-Based visible platform-- Smart Vehicles Controlling System under the assistance of the Sony Smart Eyeglass. Three devices including Intel Edison developer board, Sony Smart Eyeglass and ultrasonic sensors are employed in the system. The road condition is collected by using the sensors and processed by the Intel Edison board. Then the smart eyeglass is used to show the information with AR technology in front of the driver's eyes as an HUD form. The information transition is based on the Bluetooth module. We also accomplish a car detection system that can be embedded into the system in the future. Unfortunately, we could not finish the connection of three parts of the system for the failure of the Bluetooth. Further prospects and visions of the smart car system are also discussed.

1 INTRODUCTION

Our project aims at realizing an IoT platform with sensors, processors and displayers cooperating with each other in order to build a smart vehicles controlling system and guarantee the safety of drivers and passengers. The innovation points lie in the improved algorithm, the combination of board and sensors, displayers and sensors. Most of all, the integration of sensors and displayer counts as the most innovative part, which makes it possible to build an integral visible platform involving HUD, alarming, automatic braking, navigation and other smart functions.
In order to demonstrate our idea we focus on a simplified model, as long as our practice on simplified model get mature and having more access to hardware materials, we can expend our model to practical use.

Edison response the trend of IoT and is born for inventors, entrepreneur and consumer product designers of wearable device, helping them to quickly produce prototypes. Compared to the previous more education-oriented Galileo development board, Edison may be directly used for prototyping and production of small and medium-sized IoT solutions, which is suitable for a variety of hardware development companies and innovation team, as well as the users who aim at change particular business and industrial fields and trig IoT revolutions against traditional lifestyles.

In our project, Edison plays the role of a brain. As a powerful processor, Edison has the capability of receiving the distance information sending from the ultrasonic sensor and transmitting this information to smart mobile phone by Bluetooth module (we use Android mobile phone in order to build connections to Sony Smart eyeglass). In terms of innovation points, Edison easily realizes physical connections with variety sensors to receive and send messages from sensors and board respectively. By Bluetooth and Wi-Fi module, it builds up remote connections theoretically, realizing a wireless integration and making it possible to liberate Edison—disconnecting it from PC and settling it on vehicles together with sensors. Therefore, a prototype IoT network appears.

Originally, as an alternative to windshield, we choose the Smart eyeglass as a displayer for showing the information we suppose to demonstrate on the smart glass.
With the knowledge of this intelligent Smart Eyeglass, we consider it as not only a glass for displaying, but also an aggregation of sensors and displayers. With Bluetooth or Wi-Fi added in, the Sony Smart eyeglass itself might realize an IoT platform itself. With these thoughts, we begin with this Smart Eyeglass based project design.

We hope to build this integral visible platform: conforming functions including HUD (Head Up Display), close-distance-alarming, automatic braking, navigation and other smart functions.

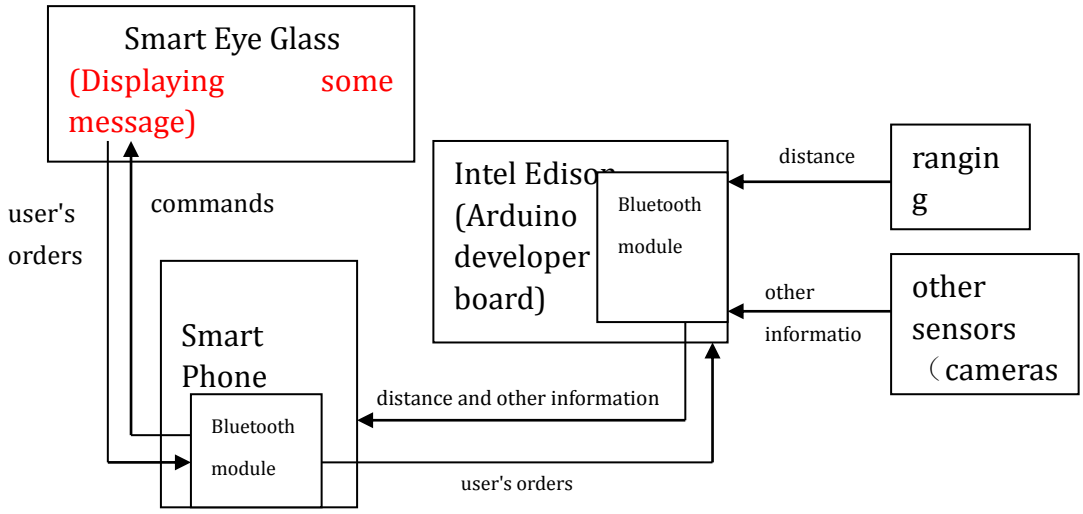
The HUD function originates from the idea of Euro-fighter, which aims at locating accurately and precisely striking the targets. Transplanting this idea to vehicles, we remove the information such as fuel, speed, distance, direction, or even date and weather from the dash board to the windshield or smart eyeglass. What is especially worth mentioning is, for sake of safety, whether the information appearing on the displayers or not and the amount of the information is up to the drivers for sure. After all, this system is designed for enhancing the safety without sacrificing too much convenience.

The combination of three essential parts: sensors, processors, and displayers—illustrates the concept of integration. Under the most ideal case, different parts get different tasks and also talk to each other, devoting to deal with real-time scenes: measuring, transmission, processing, feedback, displaying and automatic controlling.

2 METHODS

2.1 Design of the system

2.1.1 Blue print of the smart car system



2.1.2 Description of the system

In the graph above, we can see several parts of our system. On the right side is the sensor, including the ultrasonic ranging sensor and some other sensors such as cameras (we use camera as a sensor). These sensors send information to the Intel Edison developer board via Bluetooth. On the board, that information is processed so that the Smart Eyeglass can easily acquire them. Then distance and other information are delivered to the smart phone, which is used as a transit point of the glass and the board. The delivery also goes through the Bluetooth module. The Smart Eye Glass is used to display the HUD on the screen, with which car driver can see information such as road environment as well as weather and time instantly. What's more, the

glass itself can send some user's orders back, which can help the board to control the sensors.
Bluetooth connects each part in our system. It is a very efficient way of devices' communication in short distance. Furthermore, both the smart eye glass and the board can be linked to an IoT(Internet of Things) cloud service, which allows a communication on a larger scale. At last, all the devices can be connected to the car system, in which some information such as fuel and speed can be acquired.

2.2 Subsystem--Vehicle detection system

2.2.1 Introduction

Vehicle detection with smart glasses is to detect vehicles ahead in order to get enough information for the drivers in a smart car.

2.2.2 Establishment of vehicle detection system

The vehicle detection system is based on two algorithms, haar classifier with adaboost and image pre-processing. Haar classifier with adaboost is used to detect vehicles, and image pre-processing is applied to accelerate the detecting process.

(a) Algorithms description
(1) Haar Classifier+Daboost
Haar-like feature, commonly used in face detection programs, is especially suitable for the detection of rectangle objects (vehicles). It defines four types of features: edge feature, line feature, centre feature and diagonal feature. In OpenCV, a haar-like training tool is provided for instant and quick haar-like feature learning.

Adaboost is an iteration machine-learning algorithm using several weak classifiers trained by the positive and negative samples to build stronger classifier. There also exists a convenient tool in OpenCV helping to build a training data file: CarsModel.xml. The file is employed in the program to detect cars in the image.

The specific description of the algorithm can be seen in the appendix.

(2) Image pre-processing algorithm
Image pre-processing algorithm is used to accelerate the process, for by using the haar classifier with adaboost, the process of detection is very slow, although it performs very well on detection.
We firstly transformed the image into a gray scale image for the Adaboost, which requires a gray scale image to run on. Then a binaryzation process is applied on the image to increase the performance of the algorithm. At last, we perform the histogram equalization on the image, so that the dark parts and the bright parts of the image can be easily detected.

(b) Frame processing

Another way to accelerate the algorithm is to speed up the frame capturing function. In order to accomplish that, we run the detection function every two frames. The frame, where the detection function does not run, only captures one frame from the video flow, and prints the circles and squares.
Also, we found that because of the low speed of the cars movement, in the continuous two frames, the positions of one car in the image won't change much. This rule can be applied in the algorithm. In the frame that does not run the detection function, we can print the squares and circles where the previous frame prints.

(c) Error dislodging
In the experiments, some errors only occur in one separate frame, and the correct detections occur in some continuous frames. We compared two continuous frames. If something was not detected in the previous frame, and is detected in the current frame, we view it as an error and delete it from the detection.

On the other hand, we found that most of the errors seem to be very small circles, so we also delete those circles, which are too small.

(d) Function Description
The whole vehicle detection system is built based on the open source computer vision library, OpenCV.
(1) Vehicles Detector
This class implements the functions of the capturing a given video, pre-processing each frame, detecting and circling the vehicles in each frame, and displaying HUD on the screen.
Step 1: Video Capture
We used cvCapture(a video capture function in OpenCV) to capture each frame of the video. The captured video can be a prerecorded video or a real time video.

Step 2: Frame Pre-processing
cvCvtColor, we firstly changed the image into a gray-scale one. Then we set the threshold to be 50 to convert the image into a black-and-white one. At last, we resized the image and use cvSqualizeHist to implement the histogram equalization.

Step 3: Vehicles Detection
After the pre-processing, the processed frame can be used to detect vehicles. With the given classifier file, "CarsModel.xml", we can apply into the cars detection using the function cvHaarDetectObjects. This function has a serial of return value in the form of CvSeq, which can be used to calculate the vehicles' positions.

Step 4: Circling the vehicles detected
Circles and squares are then drawn around the points of vehicles' positions. Furthermore, different colors are used to show distance between the vehicle and the

vehicle in front. For example, when the circle is bigger than a threshold, we use red color to circle the vehicle in front, as a warning of danger.

(2) HUD display

We used the class HUD to display HUD on the smart eyeglass. Information on velocity, proportion of fuel, current date and time and other information are included in this class.

(e) Testing

A test video(captured in a car running on a road in Nanjing) is used to test our vehicle detection program. For most of the time, this program runs quite well, and it can at least detect one car in the frame. However, in some frames, there are some errors caused by the interruption of the environment. There are also some omissions in the detection because the training set is not big enough.

2.2.3 Results

(a) The success, error and omission rate

The table 2.2.1 shows the accuracy of our detection. We capture one frame in every 54 frames of the detection, and figure out the ratio of errors (detecting wrong objects), omissions (fail to detect the cars) and success.

Image name	Number of cars	Success	Errors	Omissions
27	5	0	0	5
135	4	1	0	3
189	4	0	0	4
243	3	2	0	1
297	2	2	0	0
351	2	0	0	2
405	2	1	0	1
459	0	0	0	0
513	1	1	0	0
567	1	0	0	1
621	2	2	0	0
675	3	2	0	1
729	2	1	0	1
783	1	0	0	1
837	1	1	0	0
891	0	0	1	0
945	2	0	0	2
999	0	0	0	0
1053	2	0	1	2
1107	2	0	0	2
1161	3	1	0	2
1215	2	1	1	1

1269	1	1	1	0
1323	2	1	0	1
1377	2	0	0	2
1431	3	2	0	1
1485	2	1	0	1
1539	2	2	0	0
1593	3	2	0	1
1647	4	1	0	3
1701	1	0	0	1
1755	1	0	0	1
1809	1	0	0	1
1863	1	0	1	1
1917	2	1	0	1
1971	2	0	0	2
2025	2	1	0	1
2079	1	1	0	0
2133	3	0	1	3
2187	1	0	1	1
2241	2	1	0	1
2295	2	1	0	1
2349	4	3	0	1
2403	3	2	0	1
2457	3	2	0	1
2511	2	0	0	2
2565	1	0	1	1
2619	1	0	0	1
2673	1	0	0	1
2727	1	0	0	1
2781	0	0	0	0
2835	0	0	1	0
2889	1	1	0	0
2943	1	0	0	1
2997	1	0	0	1
3051	1	1	0	0
3105	1	0	0	1
3159	2	1	1	1
3213	0	0	2	0
3267	2	1	1	1
3321	0	0	2	0
3375	3	2	0	1
3429	2	1	1	1
3483	2	2	0	0
3537	2	2	0	0

3591	2	1	0	1
3645	2	1	0	1
Total	120	50	16	70
Rate		41.67%	13%	58.3%

Table 2.2.1

The above table shows that the success rate is 41.67%. However, by using the techniques of pre-processing and frame processing, the rate of error is lowered to 13% . Unfortunately the omission rate is still up to 58.3%.

(b) Pictures about what the system shows

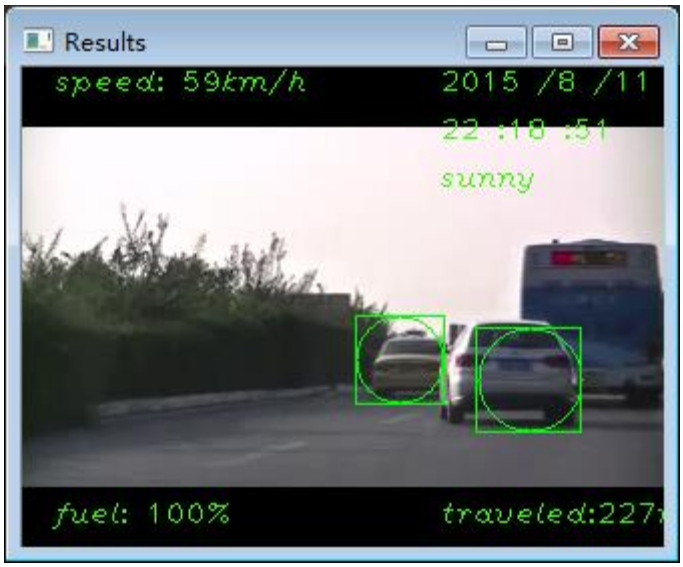


Figure 2.2.1

Normal detection model, in which green lines indicates safe distance

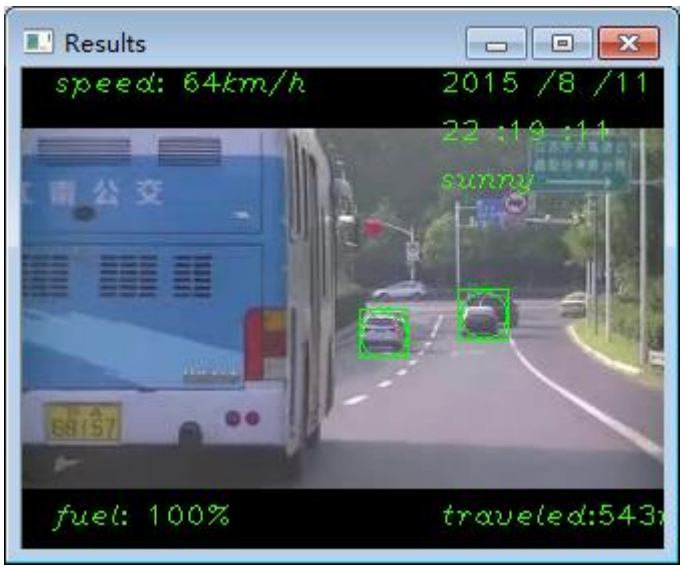


Figure 2.2.2

Longest distance from which the detection function can recognize a car

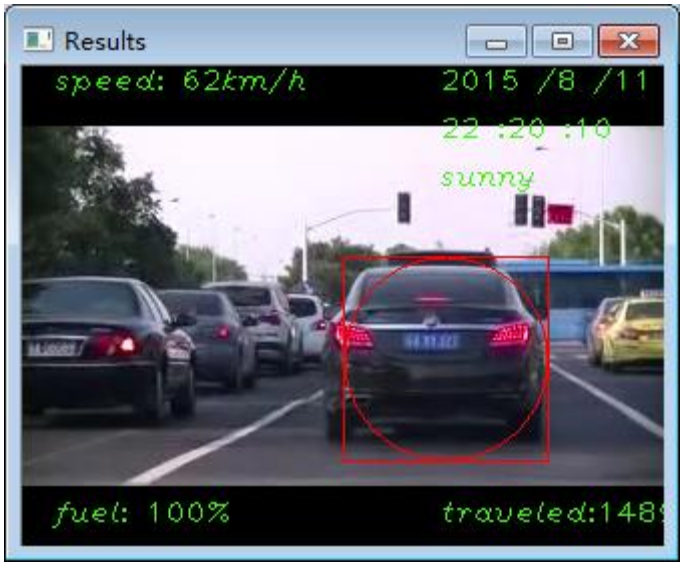


Figure 2.2.3

Red circles and squares that are used to warn the driver of the dangerous distance between two vehicles

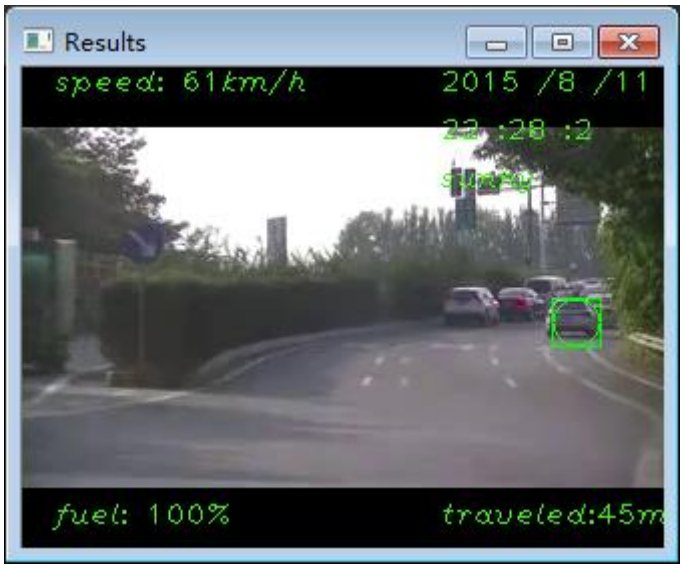


Figure 2.2.4

The HUD here shows the speed, the date, the time, the weather, the proportion of fuel and the distance the car have traveled.

The above pictures show how the detection system works, and how the HUD is displayed on the screen. In the HUD system showed in the picture, we use a presumptive speed because without a speed sensor, it is very hard to measure speed using one camera. And using the speed, we can calculate the fuel used given the fuel consumption per kilometer and the distance the car has traveled.

The test video we use can be seen in the appendix.

2.2.4 Discussions

(a) Analysis of the results

In the results of the system, we can see that the algorithm we use has the basic functions of detecting cars, but there still exists a high probability of missing the detection of cars. At the same time, the error rate is much lower than the success rate, which means the methods we have used to reduce the error rate works.
We assume that the small scale of our training set may cause the problem of high ratio of omission. What's more, the training set is not accurate enough. If we can have a larger and more accurate training set, the success rate would increase.

(b) Improvements of the algorithm
(1) Using motion detection to separate moving foreground
We tested a pre-defined function BackgroundSubtractorMOG2 in OpenCV. This function is used to separate the moving objects from the background. We put in into the pre-processing function, but unfortunately, it cannot work normally because in our case, the video is captured in a moving car, which means the background and the foreground is moving together. Using this function cannot do the separation, even when the car in front is moving at the same speed of the driver's car, the vehicle in front seems to be static, which fails the motion detecting function.
We also tested other kinds of motion detection methods such as the moving background difference method, but these methods all failed because of the moving background. We believe that we should use some more complicated motion detection methods to separate the moving objects in our further study.

(2) Using other features to detect cars
Compared to face detection, cars detection is much easier because all the cars have similar appearance: the echelon-like carriage basing on a rectangle-like chassis. This appearance feature can be used to training the classifier since nothing on the road has such shape. In some cases we have learnt, we can also detect cars by detecting the shade under the car when it is sunny. This method is important because of its convenience of programming and the fast speed, which means we could apply it on the embedded system in the future.

(3) Using HoG feature + SVM
HoG feature + SVM is another famous algorithm in human body detection. However, it seems that in the cars detection problem, HoG+SVM runs much slower than Haar+Adaboost. We assume that we should do some special pre-processing functions to accelerate the HoG feature+SVM methods. Furthermore, if this system is applied into practice, we could use this method to detect pedestrians on the road.

2.3 Subsystem----Intel Edison

2.3.1 Introduction

Edison is the latest release of general-purpose computing platform, dual-core, 500 MB, the latest 22-nanometer Atom (core computing platforms), 100 megabytes of MCU, powerful computing capability. With Arduino breakout board, people with mobile phones and other devices could connect to anyone and anything in the future.

In 2020, all devices will be able to connect to each other intelligently, whose number is up to more than 50 billion. As a computing platform, the volume of Edison is small, while the applications will widely range, such as industrial control, healthy and medical care close to people's livelihood, air testing, transportation, etc.

2.3.2 Establishment of vehicle measurement system

(a) Hardware Connection

(1) Embedded System: Intel® Edison

Intel® Edison is a very strong embedded system which can be used as a card computer. We use Intel® Edison as the center of our system because of its high quality performance. What's more, Intel® Edison has interface connecting to the sensors and it has its own Wi-Fi and Bluetooth module, enabling the transition of sensor information with the smart eye glass to be much easier.

Intel® Edison is used to get data collected from the sensors, as well as from the Internet through the Wi-Fi module. The board is also used to do some basic data processing, which can smoothen the process of transition.

(2) Sensors

Several sensors, which are distributed on the user's car, are used to acquire different information from the environment. For example, we have used an ultrasonic sensor on the head of the car to measure the distance between the driver's car and the vehicles ahead. Also, in the future, we can use some more accurate sensors to look up the weather, the temperature and other environment information the driver is interested.

The camera, which we view it as the 'eye' of the smart car system, is another important part of the sensor system. A camera can capture real time videos. Those videos can used to deal with many things, such as warning the driver if it is too close to the vehicle ahead, measuring the distance between two cars, recognizing the signs on the road to give the driver a hint. What we have accomplished is using the pre-captured video to detect vehicles.

Ultrasonic Sensor

The IO port TRIG trigger ranging to at least 10us high signal; module automatically sends eight 40khz square wave and automatically detect whether a signal return; the signal returned by IO port ECHO outputs a high level, high duration is ultrasonic time from launch to return. Test distance = (high level time * sound velocity (340M / S)) / 2. This module is simple to use, a control port to send a 10US high above, we can wait for a high output at the receiving port there. Output can open the timer, when this port goes low timer value can be read, this time for this distance of time, before calculating the distance. This constant cycle of measurement, which can reach you the value of mobile measurement.

(b) Software Support

The code includes two libraries to call functions for LCD displaying. LCD is previously set as green background, then let the ultrasonic sensor to send and receive signal, calculating the time and turning it into distance information. By serial print it appears on the screen the exact distance. With a threshold previously set, the buzzer Alarms when the distance below the threshold and the screen turns red as well to reminding.

(c) Function Description

With sensor being connected with Edison, code being written into Edison with the help of Ardrino and Edison being powered by batteries, we manage to measure the distance between remote control car and barrier (as is shown in the pictures and video as follow).

2.3.3 Results

Function Realization

Under most ideal case, the distance information should be transmitted to mobile phone by Bluetooth module of Edison. Being restricted by time and driver factors, we have not had Bluetooth module worked. To show the distance information directly, we set up a LCD module to display the real-time distance and alarm the drivers when distance gets too close by preset a threshold.

2.3.4 Discussions

(a) Results Evaluation

By setting the alarming threshold, the car can intelligently detect the distance from cars and barriers above. Connecting the LCD module, setting the alarming colour, the system can automatically appears the distance information and reminding. By appropriate remodel, the ultrasonic sensor can also realize reversing monitor. In terms of building a smart IoT platform and integral visible platform, we failed to figure out the Bluetooth module to build connections between Edison and mobile phone, resulting in the failure of transmitting data from mobile phone to Eyeglass.

(b) Errors Reflections

Due to the delay of the delivery, we have got only one single day on managing to debug the ultrasonic sensor. It remains some inaccuracy in measuring the distances. The measurement results turn out as abnormal and illogical in one number out of ten. We consider it might be due to the instability and inaccuracy itself or some problems of code. More time to debug is to be appreciated. In addition, taken the shaking and collision into consideration, the measurement in movement still remains discussion and adjustment.

(c) Improvements

With limited documents, we have to figure out how to set up connection between PC and the Edison, including the OS, drivers, putty, and line connection between different sensors and the board. Setting up and activating the Bluetooth module of

Edison is also a great challenge. We might need to update the flash and adjust the version of blueZ installation according to specific situation. As alternatives, we also need to prepare connecting iBeacon and IoT cloud, which involves Wi-Fi module of the board.

2.4 Subsystem---- Sony Smart Eyeglass

2.4.1 Introduction

Smart Eyeglass System is aimed to present the HUD information including fuel, speed, date, time and distance.

2.4.2 Establishment of Smart Eyeglass System

It constitutes BLE module and HUD module. BLE module is used to transfer data between smartphone and the Edison board, which has an advantage of presenting the size and content of the data on the smartphone. HUD module is used to present the basic information of the vehicles on the smart eyeglass.

(a) Function Description

The Smart Eyeglass System is based on the Sony Smart Eyeglass SDK, Sony Smart Eyeglass Sample and Android Sample.

(1) BluetoothLeService

This class implements the functions of the BLE, which is partly provided by Google. .

Step 1. State detection

By detecting the state, the method set the state to certain value.

Step 2. Data reading

After the connection state value is set to true, the method begins to read data repeatedly.

Step 3. Data updating

At the same time of data reading, the method update the information repeatedly.

(2) InformationDisplay

By using InformationDisplay class, which is modified from Sony Smart Eyeglass sample, we can display our information transferred via Bluetooth module from the Edison board.

2.4.3 Results

(a) BLE module.

At the same time of operating on the smartphone, the relevant information is displayed on the smart eyeglass. The BLE module has a operating window on the smartphone with button “connection” on it. When the Bluetooth is connected, the information of the connected devices will be showed in detail. And the transferred data from either terminal will be displayed in list.

(b) HUD module.

When the smartphone is connected to the smart eyeglass, the data transferred from the Edison board will be displayed on the smart eyeglass.

2.4.4 Discussion

(a) Disadvantage & Improvement of the BLE module.

(1) Limitations of distance

Because of the limitations of the BLE module itself, the distance between the board and the smartphone will not be far. In this case, we think Wifi module is a better module to transfer data.

(2) Implementation disadvantage

Because of the design itself is direct, some conflicts between the devices will appear.

(b) Disadvantage & Improvement of the HUD module.

(1) Complexity of operation

Because the smart eyeglass must be controlled by the smartphone, the user have to control the smartphone when driving. It's not convenient and dangerous for vehicle driving. We need to simplify the operation in next iteration.

(2) Irrationality of the UI

Because the limitation of the eyeglass's UI, information display is limited. Further improvement on the UI design should be made.

3 RESULTS

From the results of each subsystems above, we can see we can display a HUD system on the Smart Eye Glass, run a car detection program on the laptop using OpenCV and build a smart car model on which the ultrasonic sensor is arranged. Unfortunately, the connection of each system has failed due to the failure of the Bluetooth module on the board. However, we leave several Bluetooth interface in the Smart Eye Glass system so that the Bluetooth can be easily connected to glass in the future. On top of that, the car detection system also runs car simulation system inside, which can also display the HUD while detecting cars. The simulation system leaves some interfaces as well to link to the smart car system one day. For further study, we think we should focus on the connection of each part of our system. At the same time, some other platform such as IoT cloud can also be applied into the system as a part of the IoT world to build a better customer experience.

4. DISCUSSIONS

4.1 Technique Challenges

We have met challenges in every part of the ternary integration platform—the Edison board, the Sony Smart eyeglass and the sensors. Practically, there is neither previous researches nor develop experience about them.

(a) Edison
With limited documents, we have to figure out how to set up connection between PC and the Edison, including the OS, drivers, putty, and line connection between different sensors and the board. Setting up and activating the Bluetooth module of Edison is also a great challenge. We might need to update the flash and adjust the version of blueZ installation according to specific situation. As alternatives, we also need to prepare connecting iBeacon and IoT cloud, which involves Wi-Fi module of the board.

(b) Smart eyeglass
The function of augmented reality of Sony Smart eyeglass is a double-edged sword in terms of safety problem. First of all, the visual range is not broad enough when wearing eyeglass, drivers might need to rotate the heads left to right, causing the distraction which exists security risks. Second, for people who are nearsighted, it might be a burden for them to wear an additional eyeglass.
The original thoughts came from building a smart vehicle controlling system as well as an integral visible platform in the windshield. As is mentioned above, whether the information appearing on the displayers or not and the amount of the information is up to the drivers themselves. So the security problems can be controlled or reduced by drivers according their personal situation. Also, relative transportation departments shall issue safety regulations to standard these settings.

(c) Sensors
Figuring out and making comparison between the 3-PIN and 4-PIN ultrasonic sensors, we have to try diverse sensors connecting, and also studying and writing the corresponding code which running in Ardrino, because the example code already have in Ardrino does not work well when measuring the distances.

(d) OpenCV Based Visual Salience Algorithm
In order to process the real time video sending back by the camera, recognizing the vehicles in front of drivers, reminding the potential danger and feeding back the exact distances, we need to deal with video streams first. We adapt special frame processing methods, departing videos into frames, and then detecting every two frames. In terms of reducing the mistakes and errors, we minus every two frame and output the results.

4.2 Limitations
(a) Instability—Edison
Since it has come out for a short time, the supplementary facilities for Edison have still remained immature. Besides, the board itself has performance quite unstably. It is often the case that connection become invalid or failure. By far, we have realized the

Bluetooth pairing, while it becomes hard to continue connecting Bluetooth to the Android mobile phone, which causes it impossible for us to achieve an independent system—without Bluetooth working, the board cannot disconnect with PC, leaving the IoT platform a theoretical possibility. Still, given adequate time and repeated attempts, we will definitely work this out.

(b) Incompleteness—Algorithm
In the results of the system, we can see that the algorithm we use has the basic functions of detecting cars, but there still exists a high probability of missing the detection of cars. At the same time, the error rate is much lower than the success rate, which means the methods we use to reduce the error rate works.
We assume that the small scale of our training set causes the problem of high ratio of omission. What's more, the training set is not accurate enough. If we can have a larger and more accurate training set, the success rate will increase.

(c) Inaccuracy—Sensor
Due to the delay of the delivery, we have got only one single day on managing to debug the ultrasonic sensor. It remains some inaccuracy in measuring the distances. The measurement results turn out as abnormal and illogical in one number out of ten. We consider it might be due to the instability and inaccuracy itself or some problems of code. More time to debug is to be appreciated. In addition, taken the shaking and collision into consideration, the measurement in movement still remains discussion and adjustment.

(d) Insufficient --Eyeglass
We have succeeded in displaying words information in Eyeglass, but we have to admit that we failed to go further step to mine more functions. Basically we have not properly used the camera and sensors of the glass itself. Imagining that a driver seated in cars with eyeglass, he/she should be in first perspective observing the road conditions, which means the perfect solution should be taking advantages of the sensors and camera belonging to the eyeglass. In contrast, it will be troublesome to separate the sensors and displayers. Resulting from the Wi-Fi connection problems between Android devices and Smart glass, the lack of developing the functions of eyeglasses as well as the problems of HCI leading to security risks, there is still quite room for improvement.

4.3 Improvements

OpenCV Based Visual Salience Algorithm Improvements

(a) Using motion detection to separate moving foreground from the static background
We tested a pre-defined function BackgroundSubtractorMOG2 in OpenCV. This function is used to separate the moving objects from the background. We put in into the pre-processing function, but unfortunately, it cannot work normally because in our case, the video is captured in a moving car, which means the background and the foreground is moving together. Using this function cannot do the separation, even

when the car in front is moving at the same speed of the driver's car, the car in front seems to be static, which fails the motion detecting function.

We also tested other kinds of motion detection methods such as the moving background difference method, but these methods all failed because of the moving background. We believe that we should use some more complicated motion detection methods to separate the moving objects in our further study.

(b) Using other features to detect cars
Compared to face detection, cars detection is much easier because all the cars have similar appearance: the echelon-like carriage basing on a rectangle-like chassis. This appearance feature can be used to training the classifier since nothing on the road has such shape. In some cases we have learnt, we can also detecting cars by detecting the shade under the car when it is sunny. This method is important because of its convenience of programming and the fast speed, which means we could apply it on the embedded system in the future.

(c) Using HoG feature + SVM
HoG feature + SVM is another famous algorithm in human body detection. However, it seems that in the cars detection problem, HoG+SVM runs much slower than Haar+Adaboost. We assume that we should do some special pre-processing functions to accelerate the HoG feature+SVM methods. Furthermore, if this system is applied into practice, we can use this method to detect pedestrians on the road.

5 CONCLUSIONS

5.1 Current Situations

The current cheap HUD modules on sale normally have poor user experience. Basically they set up a screen, which reflects speed information back to the windshield. One of the most severe problems is the trouble of switching visual focus, it apparently obeys the original goal of the design—it attempts to raise the convenience with sacrificing the driving safety. Sadly, it also fails to be convenient. Under strong sunshine and when taillight is strong in the evening, it might be difficult for drivers to read the information from the screen.

On the other hand, some vehicle companies have come up with some abstract designs of HUD, but most of them are still staying in the fantasy stage, or cost too much to realize such little functional advance.

5.2 Future Foreground

In terms of the market situation mentioned above, we consider that the foreground of this visible vehicles controlling system must be promising. We have the confidence

that our project is going to have advantage over the existed market situation both on technical and economical perspective.

Owing to the advanced features of holographic projection of the smart eyeglass, we solve the inconsistent of visual focus. We hope that this improvement will have engendered the great innovation in transportation field for it supply the drivers with convenience and to the maximum extent enhance the safety.

Contributing to the high smart degree and the potential interactivity of the IoT network, our visible vehicles controlling system contains tremendous business application value, especially in this society where traffic safety becomes increasingly essential issue.

REFERENCES

1.spencer_chong.(2014). 【图像处理】Haar Adaboost 检测自定义目标（视频车辆检测算法代码）Retrieved on 13rd Aug, 2015 from:
http://blog.csdn.net/zhuangxiaobin/article/details/25476833
2.tianyahaijiao.(2011). 超声波模块 HC-SR04 简介以及编程 Retrieved on 24th August ,2011 from:
http://blog.sina.com.cn/s/blog_62efd1040100v4hn.html
3.R. Punitha, G. Suchithra, A. Sujitha (International Journal of Scientific & Engineering Research, Volume 6, Issue 1, January-2015 1542 ISSN 2229-5518) Automatic car control during heart attack with an emergency messaging and comprehensive health monitoring system
4. Andrejs Potapovs, Andrew Mor-Yaroslavtsev, Anatoly Levchenkov, Mikhail Gorobetz ()Smooth Braking of Train Using Adaptive Control Algorithms on Embedded Devices
5. Matthew Webster(2011) Mechanical Actuation and Low Level Control for a BMW X5 Automatic Safety System

APPENDIXES

1. Smart Eye Glass displaying system

Constants:

```
public final class Constants {

	public static final String EXTENSION_KEY =
			Constants.class.getPackage().getName() + ".key";

	public static final String LOG_TAG = "Start";

	private Constants() {
	}
}
```

MessageActivity:

```
import com.sonyericsson.extras.liveware.aef.registration.Registration;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;


public final class MessageActivity extends Activity {

	@Override
	public void onCreate(final Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.phonelayout);

		Button btnGlass = (Button) findViewById(R.id.btnglass);
		btnGlass.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(final View v) {
				startExtension();
			}
		});
```

```
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            String message = extras.getString("Message");
            Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_LONG)
                    .show();
        }

        if (MessageExtensionService.Object == null) {
            Intent intent = new Intent(Registration.Intents
                    .EXTENSION_REGISTER_REQUEST_INTENT);
            Context context = getApplicationContext();
            intent.setClass(context, MessageExtensionService.class);
            context.startService(intent);
        }
    }

    public void startExtension() {
        // Check ExtensionService is ready and referenced
        if (MessageExtensionService.Object != null) {
            MessageExtensionService.Object
                    .sendMessageToExtension("Get started");
        }
    }
}
```

MessageControl:

```
import android.content.Context;
import android.util.Log;
import com.sony.smarteyeglass.SmartEyeglassControl;
import com.sony.smarteyeglass.extension.util.SmartEyeglassControlUtils;
import com.sonyericsson.extras.liveware.extension.util.control.ControlExtension;
import
com.sonyericsson.extras.liveware.extension.util.control.ControlTouchEvent;

public final class MessageControl extends ControlExtension {

    private final SmartEyeglassControlUtils utils;

    private static final int SMARTEYEGLASS_API_VERSION = 1;

    public MessageControl(final Context context,
            final String hostAppPackageName, final String message) {
```

```
        super(context, hostAppPackageName);
        utils = new SmartEyeglassControlUtils(hostAppPackageName, null);
        utils.setRequiredApiVersion(SMARTEYEGLASS_API_VERSION);
        utils.activate(context);

        MessageExtensionService.Object.SmartEyeglassControl = this;

        if (message != null) {
            showToast(message);
        } else {
            updateLayout();
        }
}

public void requestExtensionStart() {
        startRequest();
}

@Override
public void onResume() {
        updateLayout();
        super.onResume();
}

@Override
public void onDestroy() {
        Log.d(Constants.LOG_TAG, "onDestroy: HControl");
        utils.deactivate();
};

@Override
public void onTouch(final ControlTouchEvent event) {
        super.onTouch(event);
        MessageExtensionService.Object
                        .sendMessageToActivity("tapping");
}

private void updateLayout() {
        showLayout(R.layout.layout, null);
        sendText(R.id.btn_update_this, "100m");
}

public void showToast(final String message) {
        Log.d(Constants.LOG_TAG, "Timeout Dialog : MessageControl");
```

```
            utils.showDialogMessage(message,
                    SmartEyeglassControl.Intents.DIALOG_MODE_TIMEOUT);
        }
}
```

MessageExtensionReceiver:

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

public final class MessageExtensionReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(final Context context, final Intent intent) {
        Log.d(Constants.LOG_TAG, "onReceive: " + intent.getAction());
        intent.setClass(context, MessageExtensionService.class);
        context.startService(intent);
    }
}
```

MessageExtensionService:

```
import java.util.List;
import android.content.Intent;
import android.util.Log;
import com.sonyericsson.extras.liveware.aef.control.Control;
import com.sonyericsson.extras.liveware.extension.util.ExtensionService;
import com.sonyericsson.extras.liveware.extension.util.ExtensionUtils;
import com.sonyericsson.extras.liveware.extension.util.control.ControlExtension;
import com.sonyericsson.extras.liveware.extension.util.registration.DeviceInfo;
import com.sonyericsson.extras.liveware.extension.util.registration.DisplayInfo;
import
com.sonyericsson.extras.liveware.extension.util.registration.RegistrationAdapter;
import
com.sonyericsson.extras.liveware.extension.util.registration.RegistrationInforma
tion;

public final class MessageExtensionService extends ExtensionService {

    public static MessageControl SmartEyeglassControl;
    public static MessageExtensionService Object;
    private static String Message = null;
```

```
public MessageExtensionService() {
        super(Constants.EXTENSION_KEY);
        Object = this;
}

@Override
public void onCreate() {
        super.onCreate();
        Log.d(Constants.LOG_TAG, "onCreate: MessageExtensionService");
}

@Override
protected RegistrationInformation getRegistrationInformation() {
        return new MessageRegistrationInformation(this);
}

@Override
protected boolean keepRunningWhenConnected() {
        return false;
}

public void sendMessageToExtension(final String message) {
        Message = message;
        if (SmartEyeglassControl == null) {
                startSmartEyeglassExtension();
        } else {
                SmartEyeglassControl.requestExtensionStart();
        }
}

public void startSmartEyeglassExtension() {
        Intent intent = new Intent(Control.Intents
                        .CONTROL_START_REQUEST_INTENT);
        ExtensionUtils.sendToHostApp(getApplicationContext(),
                        "com.sony.smarteyeglass", intent);
}

public void sendMessageToActivity(final String message) {
        Intent intent = new Intent();
        intent.setClass(getBaseContext(), MessageActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.putExtra("Message", message);
        startActivity(intent);
}
```

```
    @Override
    public ControlExtension createControlExtension(
            final String hostAppPackageName) {
        ScreenSize size = new ScreenSize(this);
        final int width = size.getWidth();
        final int height = size.getHeight();
        List<DeviceInfo> list = RegistrationAdapter.getHostApplication(
                this, hostAppPackageName).getDevices();
        for (DeviceInfo device : list) {
            for (DisplayInfo display : device.getDisplays()) {
                if (display.sizeEquals(width, height)) {
                    return new MessageControl(this,
                            hostAppPackageName, Message);
                }
            }
        }
        throw new IllegalArgumentException("No control for: "
                + hostAppPackageName);
    }
}
```

MessageRegistrationInformation:

```
import android.content.ContentValues;
import android.content.Context;
import
com.sonyericsson.extras.liveware.aef.registration.Registration.ExtensionColumns
;
import com.sonyericsson.extras.liveware.extension.util.ExtensionUtils;
import
com.sonyericsson.extras.liveware.extension.util.registration.RegistrationInforma
tion;

public final class MessageRegistrationInformation
        extends RegistrationInformation {

    private final Context context;

    private static final int CONTROL_API_VERSION = 4;

    public MessageRegistrationInformation(final Context context) {
        this.context = context;
    }
```

```
    @Override
    public int getRequiredControlApiVersion() {
        return CONTROL_API_VERSION;
    }

    @Override
    public int getTargetControlApiVersion() {
        return CONTROL_API_VERSION;
    }

    @Override
    public int getRequiredSensorApiVersion() {
        return API_NOT_REQUIRED;
    }

    @Override
    public int getRequiredNotificationApiVersion() {
        return API_NOT_REQUIRED;
    }

    @Override
    public int getRequiredWidgetApiVersion() {
        return API_NOT_REQUIRED;
    }

    @Override
    public ContentValues getExtensionRegistrationConfiguration() {
        String iconHostapp = getUriString(R.drawable.icon);
        String iconExtension = getUriString(R.drawable.icon_extension);
        String iconExtension48 = getUriString(R.drawable.icon_extension48);

        ContentValues values = new ContentValues();
        values.put(ExtensionColumns.CONFIGURATION_ACTIVITY,
                MessageActivity.class.getName());
        values.put(ExtensionColumns.CONFIGURATION_TEXT,
                context.getString(R.string.configuration_text));
        values.put(ExtensionColumns.NAME,
                context.getString(R.string.extension_name));
        values.put(ExtensionColumns.EXTENSION_KEY,
Constants.EXTENSION_KEY);
        values.put(ExtensionColumns.HOST_APP_ICON_URI, iconHostapp);
        values.put(ExtensionColumns.EXTENSION_ICON_URI, iconExtension);
        values.put(ExtensionColumns.EXTENSION_48PX_ICON_URI,
```

```
iconExtension48);
            values.put(ExtensionColumns.NOTIFICATION_API_VERSION,
                        getRequiredNotificationApiVersion());
            values.put(ExtensionColumns.PACKAGE_NAME,
context.getPackageName());
            return values;
      }

      @Override
      public boolean isDisplaySizeSupported(final int width, final int height) {
            ScreenSize size = new ScreenSize(context);
            return size.equals(width, height);
      }

      private String getUriString(final int id) {
            return ExtensionUtils.getUriString(context, id);
      }
}
```

ScreenSize:

```
import android.content.Context;
import android.content.res.Resources;

public final class ScreenSize {

      private final int width;

      private final int height;

      private final Context context;

      public ScreenSize(final Context context) {
            this.context = context;
            Resources res = context.getResources();
            width                                                                  =
res.getDimensionPixelSize(R.dimen.smarteyeglass_control_width);
            height = res.getDimensionPixelSize(
                        R.dimen.smarteyeglass_control_height);
      }

      public Context getContext() {
            return context;
      }
```

```
        public int getWidth() {
                return width;
        }

        public int getHeight() {
                return height;
        }

        public boolean equals(final int width, final int height) {
                return this.width == width && this.height == height;
        }
}
```

2. Car Detection system

HUD.h:

```
#include "cv.h"
#include "highgui.h"
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace std;
using namespace cv;

class HUD
{
public:
        int velocity;
        int fuel;
        time_t curtime;
        int cursecond;
        int presecond;
        int distanceTraveled;
        string weather;
public:
        HUD()
        {
                srand(time(0));
                velocity = 60;
                fuel = 100;
                curtime = time(0);
                cursecond = presecond = time(0);
                distanceTraveled = 0;
                weather = "sunny";
        }
        void setVelocity(int Velocity){velocity = Velocity;}
```

```
    void setFuel(int Fuel){fuel = Fuel;};
    void setCurtime(time_t t){curtime = t;}
    void setPresecond(int pre){presecond = pre;}
    void setCursecond(int cur){cursecond = cur;}
    void setDistance(int d){distanceTraveled = d;}
    void setWeather(string Weather){weather = Weather;}

    int getVelocity(){
        int r = rand()%10-5;
        return velocity+r;}
    int getFuel(){return fuel;};
    time_t getCurtime(){return curtime;}
    int getPresecond(){return presecond;}
    int getCursecond(){return cursecond;}
    int getDistance(){return distanceTraveled;}
    string getWeather(){return weather;}
};
CarsDetection.h:
#include "HUD.h"
using namespace std;
using namespace cv;
struct Cars                                                    //汽车结
构体，用于存储每一帧中被检测到的车辆信息
{
    schar* carsObjects;                                        //存储汽
车信息
    CvPoint carsPositions;                                     //以
CvPoint形式保存的汽车位置
    int weight;                                                //权重，用
于准确识别汽车
    bool isCar;                                                //检测是
否是汽车
};

class carsDetector
{
private:
    void getCarsPositions(CvSeq*, CvPoint*);                   //获取汽车位
置方法，输入检测到位置的原始数据（CvSeq*），返回对应的位置（CvPoint*)
    double getDistance(CvPoint, CvPoint);                      //获取两
点距离
    //汽车识别主要过程
private:
    IplImage* imagePreparation(IplImage* img);                 //读入帧
```

```
的预处理，返回处理好的图片
    CvSeq* carsDetection( IplImage* image );                    //汽车
追踪，返回追踪到当前帧中的汽车位置原始数据
    void printSquaresandCircle(CvSeq* objects,IplImage* img);    //输出框
和圈
    void printHUD(IplImage* image);
public:
    carsDetector(int filetype, char* filename, char* cascadename);
    ~carsDetector(){ cvClearMemStorage(storage);
    cvReleaseVideoWriter(&writer); }
    void frameCapture();                                         //读取
每一帧，并做简单分析
private:
    HUD hud;
    int detectVelocity(){return hud.getVelocity();}
private:
    CvVideoWriter *writer;
    CvMemStorage* storage;
    CvHaarClassifierCascade* cascade;                            //分类器
    double scale;                                                //缩放比
例，用于调整算法速度和精度
    char* file_name;                                            //读入视频文
件路径
    int file_type;                                               //== 0  处
理视频，== 1  处理摄像头
    //用于运动物体分离预处理
    BackgroundSubtractorMOG2 mog;                                //前景
分离器
    IplImage* foreground;                                        //检测到
的前景

    CvScalar colors[];                                           //画圈和
方框所使用的颜色

};
CarsDetection.cpp:
#include "CarsDetection.h"

void cvText(IplImage* img, string text, int x, int y)
{
     CvFont font;

     double hscale = 1.0;
     double vscale = 1.0;
```

```
        int linewidth = 0.5;
        cvInitFont(&font,CV_FONT_HERSHEY_PLAIN
|CV_FONT_ITALIC,hscale,vscale,0,linewidth);

        CvScalar textColor =cvScalar(0,255,50);
        CvPoint textPos =cvPoint(x, y);

       cvPutText(img, text.c_str(), textPos, &font,textColor);
}
string numToString(int num)
{
       ostringstream ost;
       ost << num;
         string str(ost.str());
         return str;
}
/************************
名称:carsDetector
功能：carsDetecot类构造函数，用于初始化
输入：int filetype：输入文件类型（=0视频流，=1摄像头）
         char* filename：目标文件路径
          char* cascade_name：使用分类器的训练文件
其他：无默认的构造函数
************************/
carsDetector::carsDetector(int filetype, char* filename, char* cascade_name)
{
       //初始化
       CvMemStorage* storage = 0;
       cascade = (CvHaarClassifierCascade*)cvLoad(cascade_name, 0, 0, 0 );
        if( !cascade )
        {
              fprintf( stderr, "ERROR: Could not load classifier cascade\n" );
              exit(0);
        }
       scale = 0.9;
       file_name = filename;
       file_type = filetype;
       colors[0] = CV_RGB(255,0,0);colors[1] = CV_RGB(0,255,0);
}
/************************
名称: getCarPositions
功能：用于将位置的原始数据转化成在frame中的位置
输入：CvSeq* objects：汽车位置数据（由分类器直接返回）
         CvPoint* carsPositions:返回的汽车在一帧中的位置
```

```
输出：CvPoint* carsPositions:返回的汽车在一帧中的位置
**********/
void carsDetector::getCarsPositions(CvSeq* objects, CvPoint* carsPositions)
{
        for (int i=0;i<objects->total;i++)
                {
                    CvRect* r = (CvRect*)cvGetSeqElem( objects, i );
                    carsPositions[i].x = cvRound((r->x + r->width*0.5)*scale);
                    carsPositions[i].y = cvRound((r->y + r->height*0.5)*scale);
                }
}
double carsDetector::getDistance(CvPoint point1, CvPoint point2)
{
    double d = 0;
    d =
((point1.x-point2.x)*(point1.x-point2.x))+((point1.y-point2.y)*(point1.y-point2.
y));
    return sqrt(d);
}
/**********
名称: frameCapture
功能：读取帧，每两帧调用一次carsDetection方法检测当前帧中的车辆
输入：无
输出：无
**********/
void carsDetector::frameCapture()
{


      IplImage *image;

      CvCapture* capture = 0;
      capture = cvCaptureFromFile(file_name);
      storage = cvCreateMemStorage(0);                    //视频处理初
始化

      CvSeq* preObjects=cvCreateSeq(0,100,100,storage);          //存放前
一帧的汽车位置CvSeq数据
      CvSeq* curObjects=cvCreateSeq(0,100,100,storage);          //存放当
前帧的汽车位置CvSeq数据
      CvSeq* carsObjects=cvCreateSeq(0,100,100,storage);
      CvPoint    preCarsPostitions[1000];                 //存放前一帧
汽车的位置，与preObjects对应
      CvPoint    curCarsPostitions[1000];                 //存放当前帧
```

```
汽车的位置，与curObjects对应
    for (int i=0;i<=999;i++) {
        preCarsPostitions[i].x = preCarsPostitions[i].y = 0;
        curCarsPostitions[i].x = curCarsPostitions[i].y = 0;
    }                                                          //位置数据初
始化

    CvPoint     Position;
    int ObjectsTotal;                                          //汽车捕捉初
始化


writer=cvCreateVideoWriter("CarsDetection.avi",CV_FOURCC('D','I','V','X'),25,cvS
ize(640,480));

    int l=-1;                                                  //帧计数器，每
两帧检测一次汽车位置
    if (capture)
    {
        while(1)
        {
            l++;
            if (l%2!=1)                                        //偶数帧，仅输
出图像和圈出车辆，不做识别
            {
                if( !cvGrabFrame( capture ))
                    break;
                image = cvRetrieveFrame( capture );            //视频一帧输
出

                if (l==1)                                      //在第一帧时获得
preObjects的数据
                {
                    preObjects = carsDetection( image );       //检测汽车返
回到preObjects，作为前一帧图像
                    getCarsPositions(preObjects,preCarsPostitions); //获得当前
画面中检测到的车的位置
                }
                printSquaresandCircle(preObjects,image);       //圈出车辆

            //  hud.setVelocity(detectVelocity());
                printHUD(image);

                //cvWriteFrame(writer,image);
```

```
            if( cvWaitKey( 10 ) >= 0 )
                        break;
        }
        else                                                    //奇数
帧，前后帧做比对，去除误检测点
        {
            if( !cvGrabFrame( capture ))
                    break;
            image = cvRetrieveFrame( capture );             //视频一帧输
出

            curObjects = carsDetection( image );            //检测汽车返回到
curObjects，作为后一帧图像
            getCarsPositions(curObjects,curCarsPostitions); //获得当前画
面中检测到的车的位置

            for (int i=0;i<curObjects->total;i++)
            {
                //比对前后两帧中获得车的位置，如果在后一帧中出现了一
个前一帧中没有出现的检测点，则将其删除
                bool flag = false;
                for (int j=0;j<preObjects->total;j++)
                {
                    if
(getDistance(preCarsPostitions[j],curCarsPostitions[i])<20)
                        flag = true;
                }
                if (flag==false)
                    {
                        cvSeqRemove(curObjects,i);
                     }
            }

            printSquaresandCircle(curObjects,image);        //圈出车辆

        //  hud.setVelocity(detectVelocity());
            printHUD(image);
            //cvWriteFrame(writer,image);
            if (l%27 == 0)
            {
                string filename = numToString(l)+".jpg";
                char* temp = (char*)filename.data();
                cvSaveImage(temp,image,0);
```

```
                }

                preObjects = curObjects;                    //下一次循环当前帧变为前一帧
                for (int q=0;q<=999;q++)
                    preCarsPostitions[q] = curCarsPostitions[q];//同时移动车辆位置信息
                if( cvWaitKey( 10 ) >= 0 )
                        break;
            }
        }
        cvReleaseImage(&image);

    }
        cvReleaseCapture( &capture );
        cvDestroyWindow("result");

}
/************************
名称: carsDetecion
功能：输入一帧，分别调用imagePreparation和cvHaarDetectObjects检测车辆
输入：IplImage* img:  当前检测帧
输出：CvSeq* objects:汽车位置CvSeq形式的数据
************************/
CvSeq* carsDetector::carsDetection(IplImage* img )          //汽车追踪，输入一帧，返回一组检测到的车的序列
{
    //Detect objects if any
    IplImage* small_img = imagePreparation(img);            //将取得的一帧先做图像预处理
    cvClearMemStorage(storage);
    //double t = (double)cvGetTickCount();                  //检测时间
    CvSeq* objects = cvHaarDetectObjects(small_img, cascade,storage,1.1,2, 0/*CV_HAAR_DO_CANNY_PRUNING*/, cvSize(20,20));
                                                            //用Haar分类器做检测，返回CvSeq*  形式的原始位置数据
    //t = (double)cvGetTickCount() - t;
    //printf( "detection time = %gms\n", t/((double)cvGetTickFrequency()*1000.) );
    cvReleaseImage(&small_img);
   return objects;
}
/************************
名称: imagePreparation
```

```
功能：图像预处理，转灰度图、提取前景，直方图均衡化和缩放
输入：IplImage* img: 需要做预处理的帧
输出：IplImage* small_img：经过处理的帧
**********/
IplImage* carsDetector::imagePreparation(IplImage* img)
{
        //Image Preparation
      IplImage* gray = cvCreateImage(cvSize(img->width,img->height),8,1);
     IplImage* graytemp = cvCreateImage(cvSize(img->width,img->height),8,1);
      IplImage*
small_img=cvCreateImage(cvSize(cvRound(img->width/scale),cvRound(img->he
ight/scale)),8,1);
     foreground = cvCreateImage(cvSize(img->width,img->height),8,1);

      cvCvtColor(img,gray, CV_BGR2GRAY);                    //转灰度图
     mog(Mat(gray),Mat(foreground),0.005);                  //检测运动前景
     //erode(Mat(foreground), Mat(foreground), cv::Mat());  //腐蚀操作
   //GaussianBlur(Mat(foreground),Mat(foreground),Size(3,3),0,0);//高斯滤波，
去除前景的噪声

     //Mat graymat = Mat(gray);
     //graymat = Mat(gray).mul(Mat(foreground));
     //gray = &IplImage(graymat);                           //前景与原图
相乘，提取运动物体
     //cvAddWeighted(gray,1,foreground,0.5,0,gray);         //运动前景与原图
混合
     //cvThreshold(gray,gray,50,255,CV_THRESH_TOZERO);      //二值化
      cvResize(gray, small_img, CV_INTER_LINEAR);          //调整图片大
小
     cvEqualizeHist(small_img,small_img);                  //直方图均衡

     //cvShowImage("foreground",foreground);
     //cvShowImage("afterpreparation",gray);

      //cvReleaseImage(&gray);

     return small_img;
}
/**********
名称: printSquaresandCircle
功能：根据输入的位置原始数据在帧上圈出汽车
输入：CvSeq* objects：  汽车位置原始数据
       IplImage* img：用于画图的帧
输出：无
```

```
**********/
void carsDetector::printSquaresandCircle(CvSeq* objects,IplImage* img)
{
    int distancecolor = 0;                                          //根据不同距离显示不同颜色的框

    for(int i=0;i<(objects? objects->total:0);++i)
    {
        CvRect* r= (CvRect*)cvGetSeqElem(objects,i);
        if (r->width>50&&r->height>75) distancecolor = 0; else distancecolor = 1;
        if (r->width>20 && r->height>20)
            cvRectangle(img, cvPoint(r->x*scale,r->y*scale), cvPoint((r->x+r->width)*scale,(r->y+r->height)*scale), colors[distancecolor]);
    }
                                                                    //画方框
    for( int i = 0; i < (objects? objects->total : 0); i++ )
    {
        CvRect* r= (CvRect*)cvGetSeqElem(objects,i);
        CvPoint center;
        int radius;
        center.x = cvRound((r->x + r->width*0.5)*scale);
        center.y = cvRound((r->y + r->height*0.5)*scale);
        radius = cvRound((r->width + r->height)*0.25*scale);
        if (radius>50) distancecolor = 0; else distancecolor = 1;
        if (radius>10)
            cvCircle( img, center, radius, colors[distancecolor], 1, 1, 0 );
    }
                                                                    //画圆
    //cvShowImage( "result", img );
}
void carsDetector::printHUD(IplImage* frame)
{
        string velocitytext = "speed: " + numToString(hud.getVelocity())+"km/h";
        cvText(frame,velocitytext,frame->width*0.05,frame->height*0.05);

        string fueltext = "fuel: "+ numToString(hud.getFuel())+"%";
        cvText(frame,fueltext,frame->width*0.05,frame->height*0.95);

        hud.setCurtime( time(0));
        tm tim =*localtime(&hud.curtime);
        string datetext = numToString(tim.tm_year+1900)+"/"+numToString(tim.tm_mon+1)+"/"+numToString(tim.tm_mday);
```

```
            cvText(frame,datetext,frame->width*0.65,frame->height*0.05);
            string timetext =
numToString(tim.tm_hour)+" :"+numToString(tim.tm_min)+" :"+numToString(ti
m.tm_sec);
            cvText(frame,timetext,frame->width*0.65,frame->height*0.15);

            hud.setCursecond(time(0));

      hud.setDistance(hud.getDistance()+hud.getVelocity()*1000/3600*(hud.getC
ursecond()-hud.getPresecond()));
            hud.setPresecond(hud.getCursecond());
            string distancetext = "traveled:"+numToString(hud.getDistance())+"m";
            cvText(frame,distancetext,frame->width*0.60,frame->height*0.95);

            string weathertext = hud.getWeather();
            cvText(frame,weathertext,frame->width*0.65,frame->height*0.25);
            cvShowImage("Results",frame);
}
```

test.cpp:

```
#include "CarsDetection.h"
int main()
{
      carsDetector detector(0,"test.avi","CarsModel.xml");
      detector.frameCapture();
      //system("pause");
}
```

3. Intel Edison program

UltraSonic.ino:

```
#include <Wire.h>
#include "rgb_lcd.h"
rgb_lcd lcd;
const int colorR = 0;
const int colorG = 255;
const int colorB = 0;

const int TrigPin = 12;
const int EchoPin = 11;
const int BuzzerPin = 2;
float cm;
void setup()
{
       lcd.begin(16, 2);
```

```
    Serial.begin(9600);
    pinMode(TrigPin, OUTPUT);
    pinMode(EchoPin, INPUT);
    pinMode(BuzzerPin,OUTPUT);
}
void loop()
{
     lcd.setRGB(colorR, colorG, colorB);
    //发一个10ms的高脉冲去触发TrigPin
    digitalWrite(TrigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(TrigPin, LOW);
    cm = pulseIn(EchoPin, HIGH) / 58.0; //算成厘米
    cm = (int(cm * 100.0)) / 100.0; //保留两位小数
    Serial.print(cm);
    Serial.print("cm");

    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(0,0);
    lcd.print(cm);
    lcd.print("cm");
    if (cm<=50)
    {
        Serial.print("    TOO CLOSE! WARNING!");
        digitalWrite(BuzzerPin, HIGH);
        delay(cm*8);
        digitalWrite(BuzzerPin, LOW);
        lcd.setCursor(0,1);
        lcd.setRGB(255,0,0);
        lcd.print("DANGER!");
        // delay(100);
    }
    Serial.println();
    cm = 0;
    delay(500);
}
```