ADVANCED DRIVER ASSISTANCE SYSTEM(ADAS) USING IMAGE PROCESSING

Harshal.P[#1], Kalyani.S[#2], Rampriya.V[#3], Prakash.P[#4]

[#1] *Student, Department of Information Technology*

[#2] *Student, Department of Information Technology*

[#3] *Student, Department of Information Technology*

[#4] *Assistant Professor, Department of Electronics and Communication Engineering*

[#1 #2 #3 #4] *Sri Sairam Engineering College, West Tambaram, Chennai – 600044*

Affiliated to Anna University, Tamil Nadu, India.

[#1] harshal.psingh@gmail.com [#2] poojasundar25@gmail.com [#3] rampriyakvr@gmail.com [#4] prakash.it@sairam.edu.in

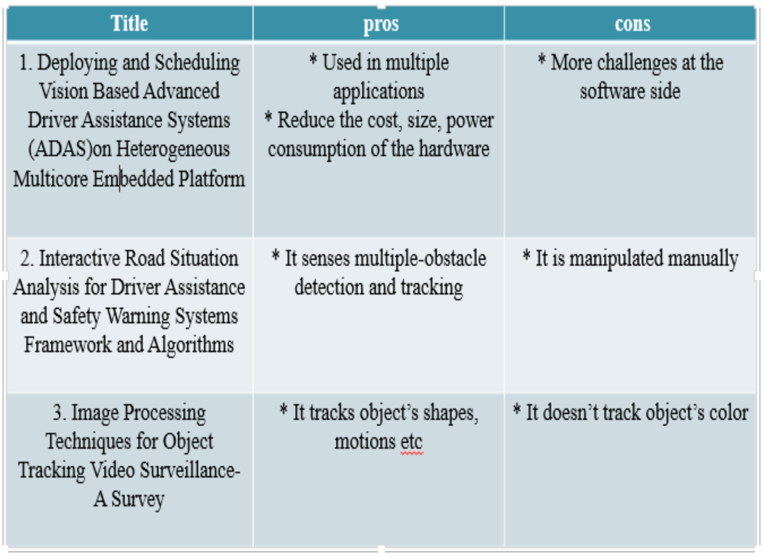
Abstract

The automotive industry can't treat safety as an afterthought. There are many innovations that have helped to reduce the number of traffic fatalities and severity of accidents. But increase in the number of vehicles on the road, more than these technologies are needed to address a further reduction of accidents and fatalities. In Advanced Driver Assistance System (ADAS) features like radar- or camera-based systems have been introduced to make driving safer. ADAS constantly keep an eye on the road and makes sure to alert the driver in real-time of an impending danger. This also detects Drowsy Driver Detection system and a traffic detection system with the external vehicle intrusion avoidance based concept. A new approach towards automobile safety and security with the autonomous region based automatic car system is proposed in this concept. We propose three distinct but closely related concepts viz. a Drowsy Driver Detection system and a traffic detection system with the external vehicle intrusion avoidance based concept. In recent time's automobile fatigue-related *crashes have really magnified. In order to minimize these issues, we have incorporated driver alert system by monitoring both the driver's eyes as well as sensing objects.*

Keywords*:Drowsy detection, intelligent vehicles, object detection,Camera-based systems.*

1. Introducion

A new approach towards automobile safety and security with the autonomous region based automatic car system is proposed in this concept. We propose three distinct but closely related concepts viz. a Drowsy Driver Detection system and a traffic detection system with the external vehicle intrusion avoidance based concept. In recent time's automobile fatigue-related crashes have really magnified. Inorder to minimize these issues, we have incorporated driver alert system by monitoring both the driver's eyes as well as sensing as well driver situation based local environment recognition based AI system is proposed. The Asian Development Bank states that “In the five years 2000–2004, more than 500 000 people were killed and around 2.6 million injured in road accidents in the People’s Republic of China (PRC), equivalent to one death every 5 minutes—the highest rate in the world.” and estimates a yearly economic loss of $12.5 billion. Driver assistance systems promise to provide partial solutions to these problems, and consequently, many research efforts aimed at developing algorithms and building frameworks forthem. Road situation analysis requires not only obstacle informationat the current time but also predicted obstacle information at the future time. Indeed, an experienced driver looks severalseconds along the road and bases his actions on information so obtained. This previewing of the road is necessary to avoid accidents since vehicle dynamics limits the car in making speedor direction changes.we proposed a road safety situation and threat analysis framework and algorithms based on driver behaviour and vehicle dynamics. In current environment modelling, obstacles are detected and tracked in future situation assessment; we use the position and size of obstacles at the current time, together with vehicle dynamics equations to predict the future road situation. For lidar data, we distinguish the object types, namely, static or moving objects, by estimating object speed. Here, a fusion algorithm is proposed to detect and track obstacles.



Title	pros	cons
1. Deploying and Scheduling Vision Based Advanced Driver Assistance Systems (ADAS)on Heterogeneous Multicore Embedded Platform	* Used in multiple applications * Reduce the cost, size, power consumption of the hardware	* More challenges at the software side
2. Interactive Road Situation Analysis for Driver Assistance and Safety Warning Systems Framework and Algorithms	* It senses multiple-obstacle detection and tracking	* It is manipulated manually
3. Image Processing Techniques for Object Tracking Video Surveillance- A Survey	* It tracks object's shapes, motions etc	* It doesn't track object's color

Figure 1. Literature survey

2.Existing System

IRsensor placing on eye for fatigue detection the problem with the system is, it is having user aiding complex with placing sensor over the eye directly.

3. Proposed System

we proposed a road safety situation and threat analysis framework and algorithms based on driver behaviour and vehicle dynamics. In current environment modelling, obstacles are detected and tracked in future situation assessment; we use the position and size of obstacles at the current time, together with vehicle dynamics equations to predict the future road situation. For lidar data, we distinguish the object types, namely, static or moving objects, by estimating object speed.
Here, a fusion algorithm is proposed to detect and track obstacles.

We have proposed the following three features in our project:
- Driver Assistance system with camera
- Vehicle external vehicle availability detection
- Human detection based attention

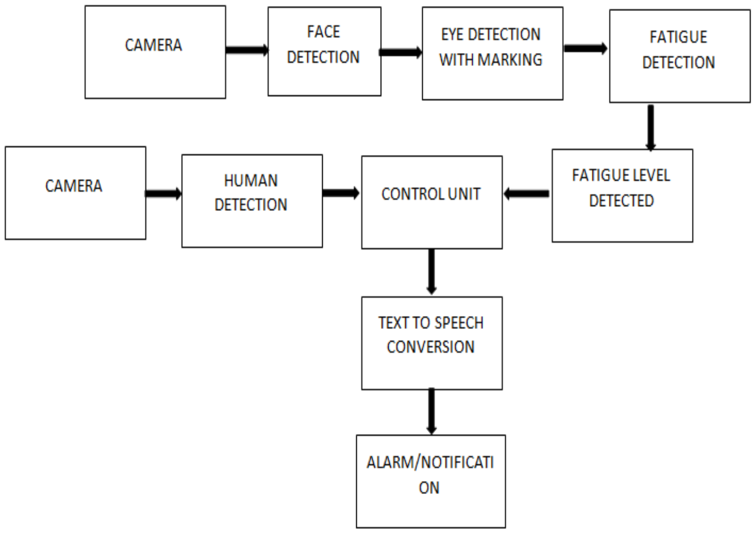


Figure.2 Block diagram of proposed system

4.System OverviewController Explanation

4.1.Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.It's capable of some amazing things, but there are a few things you're going to need to know before you plunge head-first into the bramble patch.

- ARM vs. x86

This means that the vast majority of the system's components, including its central and graphics processing units along with the audio and communications hardware, are built onto that single component is hidden beneath the 256 MB memory chip at the centre of the board.It's not just this SoC design that makes the BCM2835 different to the processor found on your desktop or laptop, however. It also uses a different instruction set architecture (ISA), known as an ARM.

The phone in your pocket almost certainly has at least one ARM-based processing core hidden away inside.The ARM-based BCM2835 is the secret of how the Raspberry Pi is able to operate on just the 5V 1A power supply provided by the onboard micro-USB port. The software for desktops and laptops is built with the x86 instruction set architecture. ARMv6 is a lightweight and powerful architecture but has a rival in the more advanced ARMv7 architecture used by the ARM Cortex family of processors. That's not to say you're going to be restricted in your choices. As you'll discover later in the book, there is plenty of software available for the ARMv6 instruction set, and as the Raspberry Pi's popularity continues to grow, that will only increase. In this book, you'll also learn how to create your own software for the Pi even if you have no experience with programming.

4.2 Windowsvs. Linux

Another important difference between the Raspberry Pi and your desktop or laptop, other than the size and price, is the operating system—the software that allows you to control the computer. The majority of desktop and laptop computers available today run one of two operating systems: Microsoft Windows or Apple OS X. Both platforms are closed source, created in a secretive environment using proprietary techniques. These operating systems are known as a closed source for the nature of their source code, the computer-language recipe that tells the system what to do. In closed-source software, this recipe is kept a closely-guarded secret. Users are able to obtain the finished software, but never to see how it's made.

The Raspberry Pi, by contrast, is designed to run an operating system called GNU/Linux—hereafter referred to simply as Linux. Unlike Windows or OS X, Linux is open source: it's possible to download the source code for the entire operating system and make whatever changes you desire. Nothing is hidden, and all changes are made in full view of the public. This open source development ethos has allowed Linux to be quickly altered to run on the Raspberry Pi, a process known as porting. At the time of this writing, several versions of Linux—known as distributions—have been ported to the Raspberry Pi's BCM2835 chip, including Debian, Fedora Remix and Arch Linux. The different distributions cater to different needs, but they all have something in common: they're all open source. They're also all, by and large, compatible with each other: software written on a Debian system will operate perfectly well on Arch Linux and vice versa. Linux isn't exclusive to the Raspberry Pi. Hundreds of different distributions are available for desktops, laptops and even mobile devices; and Google's popular Android platform is developed on top of a Linux core. If you find that you enjoy the experience of using Linux on the Raspberry Pi, you could consider adding it to other computing devices you use as well. It will happily coexist with your current operating system, allowing you to enjoy the benefits of both while giving you a familiar environment when your Pi is unavailable.

As with the difference between ARM and x86, there's a key point to make about the practical difference between Windows, OS X and Linux: software written for Windows or OS X won't run on Linux. Thankfully, there are plenty of compatible alternatives for the overwhelming majority of common software products—better still, the majority are free to use and as open source as the operating system itself.

4.3Raspberry Pi Computer Module

The Raspberry Pi Computer Module (CM1), Compute Module 3 (CM3) and Compute Module 3 Lite (CM3L) are DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (for CM1 and CM3) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these module have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards opening up more options for the designer. The CM1 contains a BCM2835 processor (as used on the original Raspberry Pi and Raspberry Pi B+ models), 512MByte LPDDR2 RAM and 4Gbytes eMMC Flash. The CM3 contains a BCM2837 processor (as used on the Raspberry Pi 3), 1Gbyte LPDDR2 RAM and 4Gbytes eMMC Flash. Finally the CM3L product is the same as CM3 except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device. Note that the BCM2837 processor is an evolution of the BCM2835 processor. The only real differences are that the BCM2837 can address more RAM (up to 1Gbyte) and the ARM CPU complex has been upgraded from a single core ARM11 in BCM2835 to a Quad core Cortex A53 with dedicated 512Kbyte L2 cache in BCM2837. All IO interfaces and peripherals stay the same and hence the two chips are largely software and hardware compatible. The pinout of CM1 and CM3 are identical. Apart from the CPU upgrade and increase in RAM the other significant hwardware differences to be aware of are that CM3 has grown from 30mm to 31mm in height, the VBAT supply can now draw significantly more power under heavy CPU load, and the HDMI HPD N 1V8 (GPIO46 1V8 on CM1) and EMMC EN N 1V8 (GPIO47 1V8 on CM1) are now driven from an IO expander rather than the processor. If a designer of a CM1 product has a suitably specified VBAT, can accomodate the extra 1mm module height increase and has followed the design rules with respect to GPIO46 1V8 and GPIO47 1V8 then a CM3 should work fine in a board designed for a CM1.

5. Features

- Low cost
- Low power
- High availability
- High reliability
- Tested over millions of Raspberry Pis Produced to date
- Module IO pins have 35u hard gold plating

6. Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1 Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1 Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1 Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

7. Software

- ARMv6 (CM1) or ARMv7 (CM3, CM3L) Instruction Set
- Mature and stable Linux software stack
- Latest Linux Kernel support
- Many drivers upstream
- Stable and well-supported userland
- Full availability of GPU functions using standard APIs

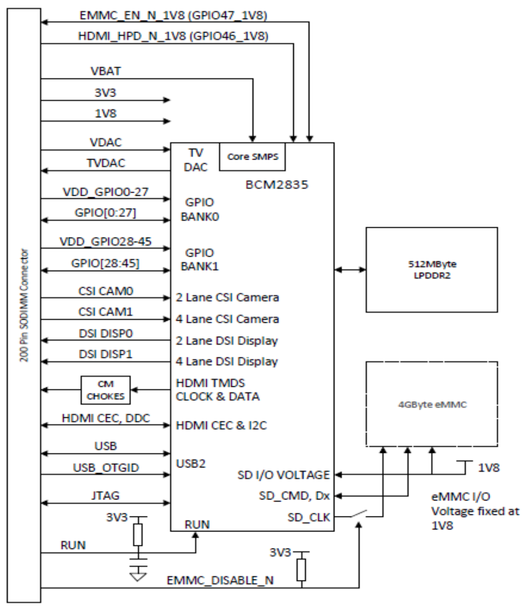


Figure3. Block Diagram OfCM1

- USB mouse
- Internet connectivity - LAN cable
- Powered USB Hub
- Case

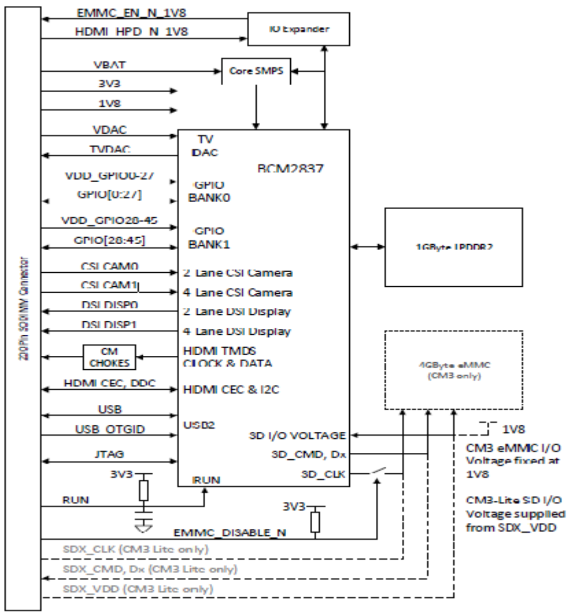
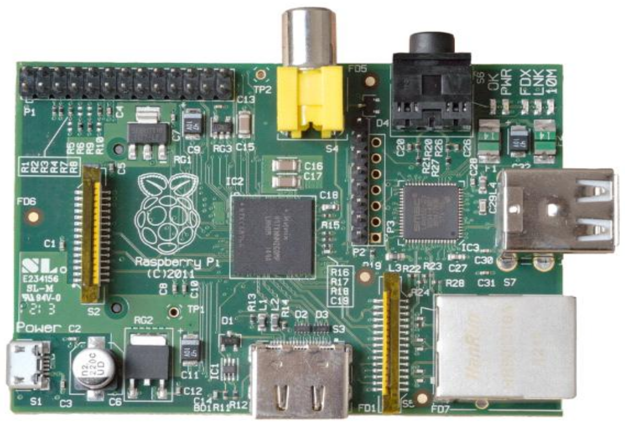


Figure4. Block Diagram CM3/CM3L

8. Features

- University of Cambridge's ComputerLaboratory:
  - Decline in skill level
  - Designed for education
- A credit card sized PC
- Plugs into a TV or monitor
- Inexpensive(ish) ~$35 each
- Capability:
  - Programming
  - Electronic Projects
  - Office
  - Play HD Videos



9. Kit components

- Raspberry Pi board
- Prepared Operating System SD Card
- USB keyboard
- Display (with HDMI, DVI, or Composite input)
- Power Supply

10. System on Chip

– A complex IC that integrates the major functional elements into a single chip or chipset.

- programmable processor
- on-chip memory
- accelerating function hardware (e.g. GPU)
- both hardware and software
- analogue components
- Benefits of SoC
- – Reduce overall system cost
- – Increase performance
- – Lower power consumption
- – Reduce size

11. SoC in Raspberry Pi: Broadcom

- BCM2835 SoC Multimedia processor
- CPU
- – ARM 1176JZF-S (armv6k) 700MHz
- – RISC Architecture and low power draw
- – Not compatible with traditional PC SoftwareGPU
- – Broadcom Video IV
- – Specialized graphical instruction setsRAM 1GB (Model 2, 3)
- – 512MB (Model B rev.2)
- – 256 MB (Model A, Model B rev.1)

12. Features

- **Chip**: Broadcom BCM2837 SoC
- **Core architecture**: ARM11
- **CPU**: 1.2 GHZ Low Power ARM1176JZFS
  Application Processor
- - Provides Open GL ES 2.0, hardware-accelerated
  OpenVG, and 1080p30 H.264 high-profile
  decode- Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs
  with texture filtering and DMA infrastructure
- **Memory**: 1 GB SDRAM
- **Operating System**: Boots from Micro SD card,
  running a version of the Linux operating system

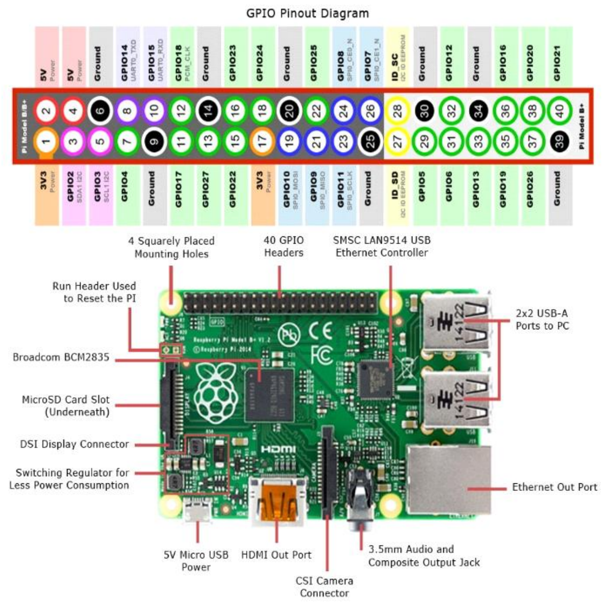


Figure 5. Pin Diagram

13.Ultrasonic Sensor

Ultrasonic sensors service the market by providing a cost effective sensing method with unique properties not possessed by other sensing technologies. By using a wide variety of ultrasonic transducers and several different frequency ranges, an ultrasonic sensor can be designed to solve many application problems that are cost prohibitiveor simply cannot be solved by other sensors. Long range detection: In industrial sensing, more and more applications require detection over distance. Ultrasonic sensors detect over long ranges up to forty feet, while limit switches and inductive sensors do not.

Broad area detection: While some photo electric sensors can detect over long distances they lack the ability to detect over a wide area without using a large number of sensors. The advantage of Migatron's ultrasonic sensors is that both wide and narrow areas can be covered. All it takes is the proper ultrasonic transducer selection. Widest range of target materials:Only ultrasonic sensors are impervious to target material composition. The target material can be clear, solid, liquid, porous, soft, wood and any color because all can be detected.

Non-contact distance measuring: Because sound can be timed from when it leaves the transducer to when it returns, distance measuring is easy and accurate to .05% of a range which equates to +or- .002 of an inch at a distance of 4 inches. It is Megatron's continuing goal to provide ultrasonic sensors in industrially hardened packages that are electrically and electronically compatible with standard controls used in today's industrial marketplace.

13.1 Advantages of Ultrasonics

When used for sensing functions, the ultrasonic method has unique advantages over conventional sensors.

- Measures and detects distances to moving objects.
- Impervious to target materials, surface and colour.

- Solid-state units have virtually unlimited, maintenance-free lifespan.
- Detects small objects over long operating distances.
- Resistant to external disturbances such as vibration, infrared radiation, ambient noise and EMI radiation.
- Ultrasonic sensors are not affected by dust, dirt or high-moisture environments.

14.Software Tools

14.1 An Introduction to Python

The Raspberry Pi gets the first half of its name from a long-standing tradition of using fruit to name new computing systems— from classic microcomputers like the Acorn, Apricot and Tangerine to more recognizably modern brands including Apple and BlackBerry—but the second half comes courtesy of the Python programming language.

14.2Introducing Python

Flexible and powerful, Python was originally developed in the late 1980s at the National Research Institute for Mathematics and Computer Science by Guido van Rossum as a successor to the ABC language. Since its introduction, Python has grown in popularity thanks to what is seen as a clear and expressive syntax developed with a focus on ensuring that code is readable. Python is a high-level language. This means that Python code is written in largely recognizable English, providing the Pi with commands in a manner that is quick to learn and easy to follow. This is in marked contrast to low-level languages, like assembler, which are closer to how the computer "thinks" but almost impossible for a human to follow without experience. The high-level nature and clear syntax of Python make it a valuable tool for anyone who wants to learn to program. It is also the language that is recommended by the Raspberry Pi Foundation for those looking to progress from the simple Scratch.

Example: Hello World
As you learned in Chapter 10, "An Introduction to Scratch", the easiest way to learn a new programming language is to create a project that prints "Hello World!" on the screen. In Scratch, you just had to drag and drop bricks of prewritten code, but in Python, you need to write this program entirely by hand. A Python project is, at heart, nothing more than a text file containing written instructions for the computer to follow. This file can be created using any text editor. For example, if you enjoy working at the console or in a terminal window, you can use nano; or if you prefer a graphical user interface (GUI), you can use Leaf pad. Another alternative is to use an integrated development environment (IDE) such as IDLE, which provides Python-specific functionality that's missing from a standard text editor, including syntax checking, debugging facilities and the ability to run your program without having to leave the editor. This chapter gives you instructions on how to create Python files using IDLE, but of course, the IDE program that you choose to use for programming is up to you. The chapter also includes instructions for running your created files directly from the terminal, which can be used in conjunction with any text editor or other IDE. To begin the Hello World project, open IDLE from the Programming menu in the Debian distribution's desktop environment. If you're not using IDLE, create a blank document in your favourite text editor and skip the rest of this paragraph. By default, IDLE opens up in Python shell mode (see Figure 11-1), so anything you type in the initial window will be immediately executed. To open a new

Python project which can be executed later, click on the File menu and choose New Window to open a blank file. The IDLE Python Shell window.

It's good practice to start all Python programs with a line known as a shebang, which gets its name from the # and !characters at the beginning of the line. This line tells the operating system where it should look for the Python files. Although this is not entirely necessary for programs that will be run from within IDLE or will call Python explicitly at the terminal, it is required for programs that are run directly by calling the program's filename. To ensure the program runs regardless of where the Python executable is installed, the first line of your program should read as follows: #!/usr/bin/env python.

This line tells the operating system to look at the $PATH environment variable—which is where Linux stores the location of files that can be executed as programs—for the location of Python, which should work on any Linux distribution used on the Pi. The $PATH variable contains a list of directories where executable files are stored, and is used to find programs when you type their name at the console or in a terminal window. To achievethe goal of printing out a message, you should use Python's print command. As its name suggests, this command prints text to an output device—by default, to the console or terminal window from which the program is being executed. Its usage is simple: any text following the word print and placed between quotation marks will be printed to the standard output device. Enter the following line in your new project:

```
print "Hello, World!"
```

The final program should look like this:

```
#!/usr/bin/env python
print "Hello, World!"
```

If you're creating the example program in IDLE rather than a plain text editor, you'll notice that the text is multi colored, (where colours are represented as differing shades of grey in the print edition). This is a feature known as syntax highlighting, and is a feature of IDEs and the more-advanced text editing tools. Syntax highlighting changes the colour of sections of the text according to their function, in order to make the program easier to understand at a glance. It also makes it easy to spot so-called syntax errors caused by forgetting to put an end-quote in a print command or forgetting to comment out a remark. For this short example, syntax highlighting isn't necessary—but in larger programs, it can be an invaluable tool for finding errors. Syntax highlighting in IDLE.

Before you run your program, save it as helloworld.py using the File menu. If you're using IDLE, the file will be given the extension .py automatically. If you're using a text editor, be sure to type .py at the end of the filename (not .txt) when you save it. This extension indicates that the file contains Python code—although Python is clever enough to run the program even if it's saved with a different file extension. How you run the file will depend on whether you're using IDLE or a text editor. In IDLE, simply choose Run Module from the Run menu, or press the F5 key on the keyboard. This will switch IDLE back to the Python shell window and run the program. You should then see the message Hello, World! appear onscreen in blue. If not, check your syntax—in particular, check that you have quotation marks at both the beginning and end of the message on the print line.

Running helloworld.py in IDLE

If you created the helloworld.py program in a text editor, you'll need to open a terminal window from the Accessories menu on the desktop. If you saved the file anywhere except your home directory, you'll also have to use the cd command to change to that directory (see Chapter 2, "Linux System Administration"). Once you're in the right directory, you can run your program by typing the following: python helloworld.py This tells the operating system to run Python and then load the helloworld.py file for execution. Unlike the Python shell in IDLE, Python will quit when it reaches the end of the file and return you to the terminal. The result, however, is the same: the message Hello, World! is printed to the standard output .

Running helloworld.py at the terminal
Making Python Programs Executable Normally, the only way to run a Python program is to tell the Python software to open the file. With the shebang line at the top of the file, however, it's possible to execute the file directly without having to call Python first. This can be a useful way of making your own tools that can be executed at the terminal: once copied into a location in the system's $PATH environment variable, the Python program can be called simply by typing its name. First, you need to tell Linux that the Python file should be marked as executable—an attribute that means the file is a program. To protect the system from malware being downloaded from the Internet this attribute isn't automatically set, since only files that are marked as executable will run. To make the helloworld.py file executable, use the chmod command (described in detail in Chapter 2, "Linux System Administration") by typing the following: chmod +x helloworld.py Now try running the program directly by typing the following:
./helloworld.py

Despite the fact that you didn't call the Python program, the helloworld.py program should run just the same as if you'd typed python helloworld.py. The program can only be run by calling it with its full location—/home/pi/helloworld.py—or from the current directory by using ./ as the location. To make the file accessible in the same way as any other terminal command, it needs to be copied to /usr/local/bin with the following command: sudocp helloworld.py /usr/local/bin/The sudo prefix is required because, for security reasons, non-privileged users cannot write to the /usr/local/bin directory. With the helloworld.py file located in /usr/local/bin, which is included in the $PATH variable, it can be executed from any directory by simply typing its name. Try changing to a different directory, and then run the program by typing the following: helloworld.py To make your custom-made programs seem more like native utilities, you can rename them to remove the .py file extension. To change the helloworld.py program in this way, just type the following line at the terminal as a single line: sudo mv /usr/local/bin/helloworld.py/usr/local/bin/helloworldOnce renamed, the program can be run simply by typing helloworld at the terminal or console.

14.3 OpenCV

OpenCV was started at Intel in 1999 by **Gary Bradsky**and the first release came out in 2000. **VadimPisarevsky**joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCVwas used on Stanley, thevehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of WillowGarage, with Gary Bradsky and VadimPisarevsky leading the project. Right now, OpenCV supports a lot of algorithmsrelated to Computer Vision and Machine Learning and it is expanding day-by-day.Currently OpenCV supports a wide variety of programming languages like C++, Python, Javaetc and is available ondifferent platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Pythonlanguage.

14.4 OpenCV-Python

Python is a general purpose programming language started by **Guido van Rossum**, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task more easier. **Numpy**is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

14.5 OpenCV-PythonTutorials

OpenCV introduces a new set of tutorials which will guide you through various functions available in OpenCV-Python.**This guide is mainly focused on OpenCV 3.x version** (although most of the tutorials will work with OpenCV 2.x also).A prior knowledge on Python and Numpy is required before starting because they won’t be covered in this guide.**Especially, a good knowledge on Numpy is must to write optimized codes in OpenCV-Python.**This tutorial has been started by *Abid Rahman K.* as part of Google Summer of Code 2013 program, under the guidance of *Alexander Mordvintsev*. **OpenCV Needs You !!!**Since OpenCV is an open source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also. So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it. And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCVin github, make necessary corrections and send a pull request to OpenCV. OpenCVdevelopers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become a open source contributor. Similar is the case with other tutorials, documentation etc.As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who knows about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV. Remember, we **together** can make this project a great success !!!

15. Contributors

Below is the list of contributors who submitted tutorials to OpenCV-Python.
1.AlexanderMordvintsev (GSoC-2013 mentor)
2.Abid Rahman K. (GSoC-2013 intern)

16. AdditionalResources

1. A Quick guide to Python - A Byte of Python
2.BasicNumpyTutorials
3.NumpyExamplesList
4.OpenCVDocumentation
5.OpenCVForum

17. Install OpenCV-Python in Windows

GoalsIn this tutorial
We will learn to setup OpenCV-Python in your Windows system. Below steps are tested in a Windows 7-64 bit machine with Visual Studio 2010 and Visual Studio 2012. The screenshotsshows VS2012.

18. Installing OpenCV from Prebuilt Binaries

1. Below Python packages are to be downloaded and installed to their default locations.
1.1.Python-2.7.x.
1.2.Numpy.
1.3.Matplotlib (*Matplotlib is optional, but recommended since we use it a lot in our tutorials*).
2. Install all packages into their default locations. Python will be installed to **C:/Python27/**.
3. After installation, open Python IDLE. Enter import numpy and make sure Numpy is working fine.
4. Download latest OpenCV release from sourceforge site and double-click to extract it.
7. Goto**opencv/build/python/2.7** folder.
8. Copy **cv2.pyd** to **C:/Python27/lib/site-packeges**.
9. Open Python IDLE and type following codes in Python terminal.

```
>>> import cv2
>>>print cv2.__version__
```

If the results are printed out without any errors, congratulations !!! You have installed OpenCV-Python successfully.

19. Building OpenCV from Source

1. Download and install Visual Studio and CMake.
1.1. Visual Studio 2012
1.2. CMake
2. Download and install necessary Python packages to their default locations
2.1. Python 2.7.x
2.2. Numpy
2.3. Matplotlib (Matplotlib is optional, but recommended since we use it a lot in our tutorials.)

Note1: In this case, we are using 32-bit binaries of Python packages. But if you want to use OpenCV for x64, 64-bit binaries of Python packages are to be installed. Problem is that, there is no official 64-bit binaries of Numpy. You have to build it on your own. For that, you have to use the same compiler used to build Python. When you start Python IDLE, it shows the compiler details. You can get more information here. So your system must have the same Visual Studio version and build Numpy from source.

Note2: Another method to have 64-bit Python packages is to use ready-made Python distributions from third-parties like Anaconda, Enthought etc. It will be bigger in size, but will have everything you need. Everything in a single shell. You can also download 32-bit versions also.
3. Make sure Python and Numpy are working fine.
4. Download OpenCV source. It can be from Sourceforge (for official release version) or from Github (for latest source).
5. Extract it to a folder, opencv and create a new folder build in it.
6. Open CMake-gui (*Start > All Programs >CMake-gui*)
7. Fill the fields as follows (see the image below):
7.1. Click on **Browse Source...** and locate the opencv folder.
7.2. Click on **Browse Build...** and locate the build folder we created.
7.3. Click on **Configure**.
OpenCV-Python can be installed in Fedora in two ways, 1) Install from pre-built binaries available in fedora repositories, 2) Compile from the source. In this section, we will see both. Another important thing is the additional libraries required. OpenCV-Python requires only **Numpy**(in addition to other dependencies, which we will see later). But in this tutorials, we also use **Matplotlib**for some easy and nice plotting purposes (which I feel much better compared to OpenCV). Matplotlib is optional, but highly recommended. Similarly we will also see **IPython**, an Interactive Python Terminal, which is also highly recommended.

20. Installing OpenCV-Python from Pre-built Binaries

Install all packages with following command in terminal as root.

```
$ yum install numpyopencv*
```

Open Python IDLE (or IPython) and type following codes in Python terminal.

```
>>> import cv2
>>> print cv2.__version__
```

If the results are printed out without any errors, congratulations !!! You have installed OpenCV-Python successfully. It is quite easy. But there is a problem with this. Yum repositories may not contain the latest version of OpenCValways. For example, at the time of writing this tutorial, yum repository contains while latest OpenCV version is With respect to Python API, latest version will always contain much better support. Also, there may be chance of problems with camera support, video playback etc depending upon the drivers, ffmpeg, gstreamer packages present etc. So my personnel preference is next method, i.e. compiling from source. Also at some point of time, if you want to contribute to OpenCV, you will need this.

21. Installing OpenCV from source

Compiling from source may seem a little complicated at first, but once you succeeded in it, there is nothing complicated. First we will install some dependencies. Some are compulsory, some are optional. Optional dependencies, you canleave if you don’t want.

22. Compulsory Dependencies

We need **CMake**to configure the installation, **GCC** for compilation, **Python-devel**and**Numpy**for creating Python extensions etc.

yum install cmake
yum install python-develnumpy
yum install gccgcc-c++

Next we need **GTK** support for GUI features, Camera support (libdc1394, libv4l), Media Support (ffmpeg, gstreamer)
etc.

yum install gtk2-devel
yum install libdc1394-devel
yum install libv4l-devel
yum install ffmpeg-devel
yum install gstreamer-plugins-base-devel

23. Optional Dependencies

Above dependencies are sufficient to install OpenCV in your fedora machine. But depending upon your requirements, you may need some extra dependencies. A list of such optional dependencies are given below. You can either leave it or install it, your call :)

OpenCV comes with supporting files for image formats like PNG, JPEG, JPEG2000, TIFF, WebP etc. But it may be a little old. If you want to get latest libraries, you can install development files for these formats.

yum install libpng-develyum install libjpeg-turbo-devel
yum install jasper-devel
yum install openexr-devel
yum install libtiff-devel
yum install libwebp-devel

Several OpenCV functions are parallelized with **Intel’s Threading Building Blocks** (TBB). But if you want to enable it, you need to install TBB first. (Also while configuring installation with CMake, don’t forget to pass -D
WITH_TBB=ON. More details below.)

yum install tbb-devel

OpenCV uses another library **Eigen** for optimized mathematical operations. So if you have Eigen installed in your system, you can exploit it. (Also while configuring installation with CMake, don’t forget

to pass -D WITH_EIGEN=ON.More details below.)
yum install eigen3-devel.

24.Conclusion

Using vibrotactile stimuli with temporal coding for conveying directional information with only two tractors is feasible and achieves an angular resolution sufficient for navigating in an interactive manner. Among the stimulus concepts tested, a code based on joint variation of modulation frequency and modulation duty cycle was associated with good performance and suitable for continuous stimulation, especially if the individual stimulus-to-response relationship was accounted for. For singular events, a code relying on the temporal shift between vibration epochs on the two sides also showed good results after participants have been informed about the rationale.In this paper we highlighted the advantage of integrating vision based Advanced Driver Assistance Systems (ADAS).We also showed that the current trends are to run such systems on heterogeneous multi-core platforms.We have provided insight of ADAS such as front camera, surround view and camera monitor systems.ADAS applications require high-performance, low power and low area solutions.

25.References

[1] P. S. Rau, “Drowsy drivers detection and warning system for commercial vehicle drivers: Field proportional test design, analysis, and progress”, Proc. - 19th International Technical Conference on the Enhanced Safety of Vehicles, Washington, D.C., 2005

[2] United States Department of Transportation., “Saving lives through advanced vehicle safety technology” http://www.its.dot.gov/ivi/docs/AR2001.pdf.

[3] Y. Takei, Y. Furukawa, “Estimate of driver’s fatigue through steering motion,” in Man and Cybernetics, IEEE International Conference, Volume: 2, pp. 1765- 1770 Vol. 2.2005

[4] W.A. Cobb., “Recommendations for the practice of clinical neurophysiology.,” Elsevier, 1983.

[5] K. Hong, Chung, “Electroencephalographic studyof drowsiness in simulated driving with sleep deprivation”, International Journal of Industrial Ergonomics, Volume 35, Issue 4, April 2005, pp. 307-320.