A Graduation Project Report Submitted to

Faculty of Engineering Cairo University

Natural and Fully Multimodal Interaction with the Vehicle and its Surroundings

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in

Electronics and Electrical Communications Engineering Department

Faculty of Engineering, Cairo University Giza, Egypt

Under the Supervision of Associate

Prof. Hassan Mostafa & Prof. Mohsen Rashwan

Submitted by:

Doaa Hussein Mohammed

Esraa Mohammed Abdelfattah

Mahmoud Ahmed Mohamed

Moataz Mohamed Nasr

Omar Talal AbdElaziz

Samar Imbaby Ismail

Academic Year 2019-2020

Chapter 1
Introduction

This thesis is submitted to the Electronics and Electrical Communications Engineering Department Faculty of Engineering, Cairo University Giza, Egypt for the Degree of Bachelor of Science. The thesis is equivalent to 20 weeks of full-time studies.

Contact Information:

Authors:
Doaa Hussein Mohammed
Esraa Mohammed Abdelfattah
Mahmoud Ahmed Mohamed
Moataz Mohamed Nasr
Omar Talal AbdElaziz
Samar Imbaby Ismail
Department of Electronics and Electrical Communications Engineering
E-Mail: doaahussein831@gmail.com
E-Mail: Mahmoudelzeiny96@gmail.com

Mentor advisor:
Mohammed Abdou
Senior Algorithms Engineer and Deep Learning Researcher at Valeo
E-mail: Mohammed.abdou@valeo.com

Abdelrahman Hussein
Teaching Assistant in Computer engineering, Cairo University
E-mail: abdelrahman.sobeih@gmail.com

University Supervision:
Name: Dr. Hassan Mostafa
E-mail: hassanmostafahassan@gmail.com
Department of Electronics and Electrical Communications Engineering Cairo University

Name: Dr. Mohsen Rashwan
E-mail: mrashwan@rdi-eg.com
Department of Electronics and Electrical Communications Engineering Cairo University

Abstract

In vehicles a new control system technology it’s called "Natural Interaction", which allows drivers to use Natural approaches to interact with their vehicle. "Customers should be able to communicate with their intelligent connected vehicle in a totally natural way," Christoph Grote, Senior Vice President, BMW Group Electronics said in a statement.

Natural Interaction technology allows the driver to combine voice, gesture, and gaze to interact with their vehicle. The driver decides how they want to interact with the vehicle by using either voice or gestures, based on their personal preferences or the current situation.

So, this project aims to provide a safer way for the driver to interact with the vehicle to prevent distractions during driving and to facilitate the gathering of surrounding information.

Keywords: Natural Interaction, Gestures Control, Voice Commands, Gaze Tracking.

Acknowledgments

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

First, we want to thank our major advisor Dr. Hassan Mustafa and Dr. Mohsen Rashwan, it has been a great pleasure and honor being our supervisors. You were continuously encouraging us, even before we decided to work on this project.

We want to thank Engineer Mohammed Abdou, our mentor from Valeo and Engineer Abdelrahman Hussein, Teaching Assistant in Computer engineering, Cairo University, for providing their time and experience to help us overcome some obstacles we faced during some stages.

Finally, we want to thank our families for their support, tolerance, and love during this year especially during the hard times they were always there having faith in what we do. We are grateful to our families, colleagues, and friends for always motivating us, without them we wouldn't have come so far.

Glossary

- **Gaze** – When someone lays eyes on you and keeps looking, it is a gaze or a stare.
- **Drowsiness** – Ready to fall asleep.
- **Gesture** – A movement usually of the body or limbs that expresses or emphasizes an idea, sentiment, or attitude.
- **Kinect** – A 3D camera that allows skeleton tracking with depth perception.
- **User** – A user here might represent a subject who is using our application with Kinect. A user here can be either a male or a female.
- **Postures** – The postures or dynamic poses are a set of movements which are performed by the user to get results and recommendation regarding their joints and weak links.
- **Assistant** – A voice assistant is a digital assistant that uses voice recognition, natural language processing, and speech synthesis to provide aid to users through phones and voice recognition applications.
- **Dropout layer** – A regularization technique used to minimize overfitting by reducing the complexity of the model where randomly selected neurons are ignored during training, where they are “dropped-out” randomly.
- **Dense layer** – A classic fully connected neural network layer where each input node is connected to each output node of the previous layer.
- **Softmax layer** – A type of squashing function limiting the output of the function into the range 0 to 1, where this allows the output to be interpreted directly as a probability, so they are used in determining the probability of multiple classes at once.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AI	Artificial Intelligence.
ANN	Artificial Neural Network.
API	Application Programming Interference.
ARM	Advanced RISC Machine.
BVLC	Berkeley Vision and Learning Center.
CNN or ConvNet	Convolutional Neural Network.
Conv	Convolution.
CPM	Convolutional Pose Machine.
CPU	Central Processing Unit.
CUDA	Compute Unified Device Architecture.
DTW	Dynamic time warping.
EAR	Eye Aspect Ratio.
etc	et cetera.
FPS	Frame Per Second.
GPU	Graphics Processing Unit.
gTTS	google Text To Speech.
iOS	iPhone OS.
MFCCs	Mel-frequency cepstral coefficients.
MNIST	Modified National Institute of Standards and Technology.

ML	Machine Learning.
MLP	Multi-Layer Perceptron.
NN	Neural Network.
NORB	NYU Object Recognition Benchmark.
L4T	Linux for Tegra.
LDA	Linear Discriminant Analysis.
LFW	Labeled Faces in the Wild.
LSTM	Long short-term memory.
OpenCV	Open Source Computer Vision.
OpenNI	Open Natural Interaction.
OS	Operating System.
PCA	Principle Component Analysis.
ReLU	Rectified Linear Unit.
ResNet	Residual Neural Network.
RGB	Red Green Blue.
SDK	Software Development Kit.
SIANN	Space Invariant Artificial Neural Networks
SSD	single-shot detector.
TPU	Tensor Processing Unit.
URL	Uniform Resource Locator.

Chapter 1.
Introduction

In this thesis, we will propose an approach to implement a natural interaction system with a vehicle that allows the driver to use voice, gesture, and gaze in various ways to interact with their vehicle to make the driving experience more productive and safer.

1.1. Motivation

“Computer vision and machine learning (ML) have started to take off, but for most people, the whole idea of what a computer is seeing when it’s looking at an image is relatively obscure.” Mike Kreiger

If you're talking about "real" artificial intelligence (AI), it's much more than just "If." The development of artificial intelligence has historically been divided into two areas; Symbolic artificial intelligence and machine learning.

Symbolic Artificial Intelligence is the field where intelligent systems are artificially designed with an if-else logic. Programmers will try to identify every possible scenario for the system to deal with. Until the late 1970s, this was the predominant form of developing the AI system. Experts in this field have strongly argued that machine learning will never last and that AI can only be written in this way.

In the last few years, Artificial intelligence and deep learning make great impacts on the automotive industry and become the new keys to success in this industry from enabling autonomous and semi-autonomous vehicles to providing advanced capabilities to several processes like design and manufacturing.

These advanced capabilities, coupled with rising consumer expectations and provided new technology for manufacturers to reduce costs and give drivers more of what they want.

AI is defined as the ability of a computer program or machine to think, learn and make decisions so AI aim is to create an intelligent system can automatically extract features and recognize a particular pattern and after having trained with datasets it can learn how to recognize this particular pattern and also take different actions, so intelligent vehicles would bring several benefits in the social, environmental, and economical fields, it able to estimate the driving scenario and react in case of danger which eliminates up to 90% of traffic accidents that are caused by human errors and saving human lives, also these vehicles can have some systems used for entertainment purposes and making the driving experience more productive such as voice assistants and ability to get information from the internet or any network.

1.2. Problem statement

Thousands of people across the world are losing their lives due to car accidents and road disasters every year and the related lost costs due to medical expenses and general maintenance and repairs costs of the road and highway systems are very large. Another big problem we face because of accidents is the traffic crowd, many people lose their time on roads due to car accidents.

The most common reasons for these accidents are human errors and distracted driving which lead to the need for a safer way for the driver to interact with the vehicle to prevent distractions while driving and to facilitate the gathering of surrounding information. This could decrease the rate of road accidents and the need to rely on manual interactions.

With AI, we get computer programs and machines to do what humans do. We feed these programs and machines with an enormous amount of data that is analyzed and processed for logical reasoning and ultimately mimic human actions.

One of the difficult problems in implementing AI systems is to save human lives on the road by eliminating or reducing unnecessary actions that a person takes that can distract him. It should also be used as an alarm that warns the driver of any near or expected danger.

1.3. Solution approach

The project aims to use Natural interaction including sound, gestures and gazes to help make using the full vehicle options of the driver easier and safer with less distraction while driving which leads to safer driving and fewer accidents, so the driver is now safer and more comfortable.

At times when the driver needs to keep his eyes on the road, the system can respond to sound and gestures for safety. This helps facilitate using the full options of the car for a driver more easily and safely with less dispersion while driving leading to a safer drive.

The use of outside interaction helps in gathering more beneficial information with the help of Machine learning models and making useful decisions based on them which helps in improving the society.

Natural interaction consists of the following main blocks:

1.3.1. Facial Recognition

One use of the system is driver authentication. Once a driver enters the vehicle, the front-facing camera snaps a photo. If the driver is recognized, their personalized data is synced to the vehicle. If they are not recognized, a notification is sent to the primary owner of the vehicle.

1.3.2. Speech Recognition

This module provides a common interface between the driver and his vehicle using speech recognition that converts the spoken words into an electronic signal that can be categorized and processed into action using devices like microphones, a special algorithm is used to establish the most likely word(s) that match the given spoken word to understand the voice command which the driver said, then do the action.

1.3.3. Eye Movement-Gaze detection

The camera focuses on one or both eyes and records eye movement, then the aspect ratio (EAR) is measured directly, and this feature is used to make sure that the driver is awake to prevent accidents.

1.3.4. Gesture direction Detection

Arm Joints tracking is used to detect the angel of action, and the place where the driver is pointing. Users can use their gestures and arms to interact and control devices without the need to touch them physically. The camera can capture joint and hand in three dimensions throughout the driver's entire operating environment and determine a precise directional vector.

1.4. Organization

The following chapters discuss the approaches have been used to implement each part of the interaction system by software, the way that we follow to integrate all subsystems and the simulation of the overall system on the hardware, so this thesis is organized as follows:

Chapter 2 explains the background of this project and gives examples of some related works to it.

Chapter 3 provides the implementation of the Face Recognition and Eye tracking subsystems and how they are designed for security purposes as driver identification and keeping his eyes on the road, also provides an explanation of the algorithms which have been used and estimate the performance and accuracy of this subsystem.

Chapter 4 explains the different models used to implement the voice recognition system and how this system is used to give the driver the ability to interact with the vehicle using several voice commands.

Chapter 5 explains the Intelligent Personal Assistant system and how it supports the internal interaction and external interaction with the driver and giving him information from different sources.

Chapter 6 provides a discussion on the different approaches used to implement the gesture control subsystem, explains the model of each approach, its advantages, disadvantages, and clarifies the method used to calculate the angle which the driver makes when pointing to the objects during the external interaction.

Chapter 7 explains the method used to integrate all subsystems to support both internal and external interaction with the vehicle.

Chapter 8 provides the results of each block and the whole system concluding the accuracy and performance.

Chapter 9 provides a brief conclusion of the system and suggestions for future work.

Chapter 2.
Background and Related work

This Chapter Contains the Necessary Background and Knowledge for the Reader to understand the following chapters.

2.1. Convolutional Neural Networks

A convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. [1][2] They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

CNNs are regularized versions of multilayer perceptron's. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. [3] Therefore, on the scale of connectedness and complexity, CNNs are on the lower extremity.

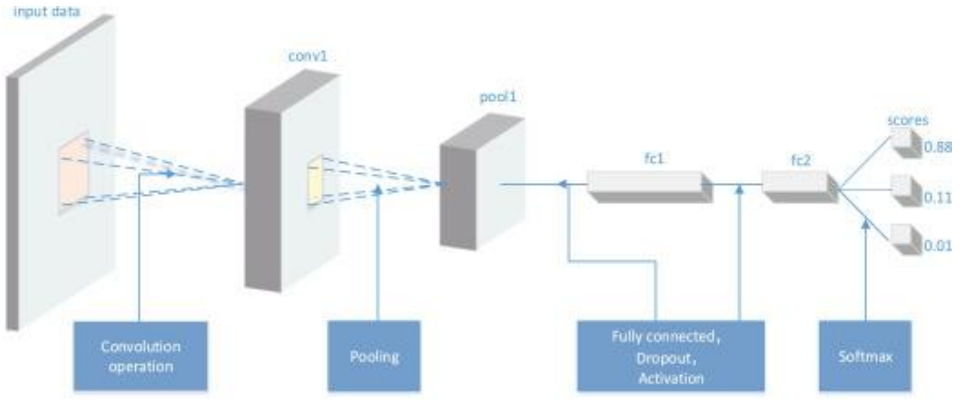

Figure 1: Convolutional Neural Networks

2.1.1. Convolutional

When programming a CNN, the input is a tensor with shape (number of images) x (image width) x (image height) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map width) x (feature map height) x (feature map channels). [4]

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. [5] A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. [6] The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. [7]

2.1.2. Pooling

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. Also, pooling may compute a max or average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer. [8]

2.1.3. Fully Connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

2.1.4. ResNet

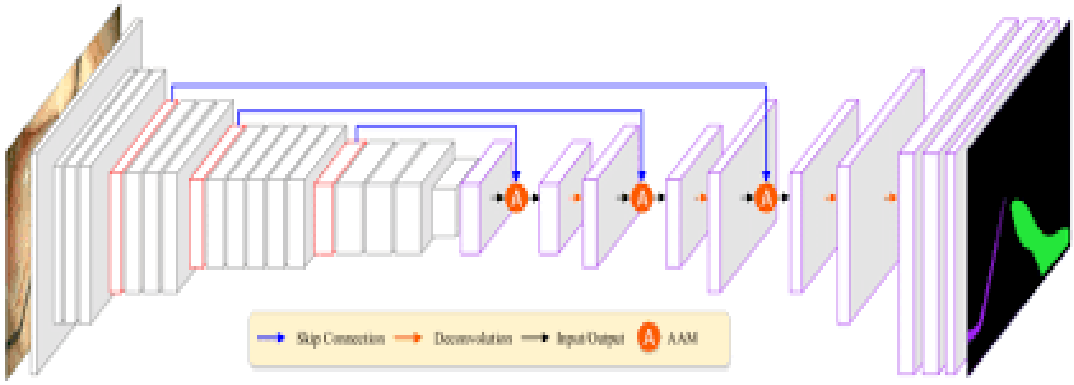

Figure 2: ResNet Architecture

A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. An additional weight matrix may be used to learn the skip weights; these models are known as Highway Nets. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a plain network. [9]

One motivation for skipping over layers is to avoid the problem of vanishing gradients, by reusing activations from a previous layer until the adjacent layer learns its weights. During training, the weights adapt to mute the upstream layer and amplify the previously-skipped layer. In the simplest case, only the weights for the adjacent layer's connection are adapted, with no explicit weights for the upstream layer. This works best when a single nonlinear layer is stepped over, or when the intermediate layers are all linear. If not, then an explicit weight matrix should be learned for the skipped connection (a Highway Net should be used). Skipping effectively simplifies the network, using fewer layers in the initial training stages. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Towards the end of the training, when all layers are expanded, it stays closer to the manifold and thus learns faster.

2.1.5. Applications

2.1.5.1. Image recognition

CNNs are often used in image recognition systems. In 2012 an error rate of 0.23 percent on the MNIST database was reported. Another paper on using CNN for image classification reported that the learning process was "surprisingly fast"; in the same paper, the best-published results as of 2011 were achieved in the MNIST database and the NORB database. Subsequently, a similar CNN called AlexNet won the ImageNet Large Scale Visual Recognition Challenge 2012. [10]

When applied to facial recognition, CNNs achieved a large decrease in error rate. Another paper reported a 97.6 percent recognition rate on "5,600 still images of more than 10 subjects". CNNs were used to assess video quality objectively after manual training; the resulting system had a very low root mean square error. [11]

2.1.5.2. Video analysis

Compared to image data domains, there is relatively little work on applying CNNs to video classification. Video is more complex than images since it has another (temporal) dimension. However, some extensions of CNNs into the video domain have been explored. One approach is to treat space and time as equivalent dimensions of the input and perform convolutions in both time and space. Another way is to fuse the features of two convolutional neural networks, one for the spatial and one for the temporal stream. Long short-term memory (LSTM) recurrent units are typically incorporated after the CNN to account for inter-frame or inter-clip dependencies. Unsupervised learning schemes for training Spatio-temporal features have been introduced, based on Convolutional Gated Restricted Boltzmann Machines and Independent Subspace Analysis. [12]

2.2. TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. [13]

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015. TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs, and without API backward compatibility guarantee: C++, Go, Java, JavaScript, and Swift (early release). Third-party packages are available for C#, Haskell, Julia, MATLAB, Scala, Rust, OCaml, and Crystal.

2.3. Face Recognition

4.2.1. OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.[14]

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of the user community and an estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups, and governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many starts ups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

4.2.2. Dlib

Dlib is a general-purpose cross-platform software library written in the programming language C++.[15] Its design is heavily influenced by ideas from a design by contract and component-based software engineering. Thus, it is, first and foremost, a set of independent software components. It is open-source software released under a Boost Software License.

Since development began in 2002, Dlib has grown to include a wide variety of tools. As of 2016, it contains software components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many other tasks. In recent years, much of the development has been focused on creating a broad set of statistical machine learning tools, and in 2009 Dlib was published in the Journal of Machine Learning Research. Since then it has been used in a wide range of domains.

2.4. Gesture Recognition

2.4.1. Kinect Sensor

Kinect is a set of motion sensors that was first produced and released by Microsoft in 2010. The technology includes a set of devices originally developed by PrimeSense, which includes RGB cameras, infrared displays, and depth detection detectors with either regulated light or time calculation Journey, a microphone suite, along with Microsoft software and artificial intelligence) to allow the device to perform real-time gesture recognition, speech recognition, and body skeletons of up to four people, among other capabilities. This enables Kinect to be used as a regular user interface without using hands to interact with the computer system. Kinect is a peripheral device that sits above the user's webcam-style screen. [16]

Kinect originated as a means to eliminate the game controller from Microsoft's Xbox video game hardware, competing with the Nintendo Wii's own motion-sensing capabilities, hoping to draw a larger audience beyond traditional video game players to the Xbox. Kinect first launched as an add-on for the Xbox 360 in November 2010, and within a few months, more than 10 million units had been sold, making it one of the fastest-selling computer hardware products at the time. However, video games had to be developed to specifically incorporate the Kinect's features, and the bulk of games released with Kinect support were family-friendly titles.

Figure 3: Kinect V2

2.4.2. libfreenect2

Libfreenect2 is an open-source cross-platform driver for Kinect for Windows.[17]

It has the following features:

- Color image processing
- IR and depth image processing
- Registration of color and depth images
- Multiple GPU and hardware acceleration implementations for image processing.

2.4.3. OPENNI2

OpenNI or Open Natural Interaction is an industry-led non-profit and open-source software project that focuses on adopting and improving interoperability for natural user interfaces and organic user interfaces for natural interaction devices (NI) and applications that use those devices and middleware that facilitate the access and use of such devices.[18]

OpenNI2 is an open-source software development kit for the RGB-D sensor primarily developed by PrimeSense Inc. OpenNI2 is still widely used in many RGB-D sensors even after development is over by PrimeSense Inc. Because OpenNI2 can be used for commercial purposes and works on cross-platform.

2.4.3.1. OpenNI Framework

The OpenNI framework provides a set of open-source APIs. These APIs aim to become a standard for apps to access natural reaction devices. The API framework itself is sometimes referred to as the OpenNI SDK.

The APIs support:

- Voice command recognition.
- Hand gestures.
- Body movement tracking.

2.4.4. NITE2

NITE2 is an OpenNI2 based broker developed by PrimeSense Inc. It has features like detecting people, positioning, hand tracking, and gesture detection. [19] NiTE2 is a closed source and its distribution has already ended. However, you can use the already distributed NiTE2.

2.4.5. OpenPose

OpenPose represents the first real-time multi-person system to jointly detect human body, hand, facial, and foot key-points (in total 135 key-points) on single images written in C++ using OpenCV and Caffe, authored by G. Hidalgo, Z. Cao, T. Simon, S.E. Wei, H. Joo, and Y. Sheikh (Robotics Institute of Carnegie Mellon University). The code has been released for full reproducibility and it is maintained and developed by the authors with the help of an active community of contributors on GitHub. [20]

2.4.6. OpenPose Features

2.4.6.1. Functionality:

- **2D real-time multi-person key-points detection:** [21][22]
  - 15, 18 or, 25-keypoint body/foot key-points estimation. Running time-invariant to the number of detected people.
  - 6-keypoint foot key-points estimation. Integrated with the 25-keypoint body/foot key-points detector.
  - 2x21-keypoint hand key-points estimation. Currently, running time depends on the number of detected people.
  - 70-keypoint face key-points estimation. Currently, running time depends on the number of detected people.
- **3D real-time single-person key-points detection:**
  - 3-D triangulation from multiple single views.
  - Synchronization of Flir cameras handled.
  - Compatible with Flir/Point Grey cameras, but provided C++ demos to add your custom input.

- **Calibration toolbox:** Easy estimation of distortion, intrinsic, and extrinsic camera parameters.
- **Single-person tracking:** for more speed or visual smoothing.

2.4.6.2. Input:

Image, video, webcam, Flir/Point Grey, and IP camera. Included C++ demos to add your custom input.

2.4.6.3. Output:

Basic image + keypoints display/saving (PNG, JPG, AVI ...), keypoints saving (JSON, XML, YML ...), and/or keypoints as array class.

2.4.6.4. Operating System:

Ubuntu (14, 16), Windows (8, 10), Mac OSX, Nvidia TX2.

2.5. Voice Recognition

2.5.1. SoX Library

SoX is a cross-platform (Windows, Linux, MacOS X, etc.) command-line utility that can convert various formats of computer audio files into other formats. It can also apply various effects to these sound files, and, as a bonus, SoX can play and record audio files on most platforms. [23]

2.5.2. FFmpeg Package

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter, and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community, or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations.[24]

2.5.3. Speech Recognition Package

Library for performing speech recognition with support for several engines and APIs, online and offline, Google Speech Recognition, Wit.ai, IBM Speech to Text, and AT&T Speech to Text.

2.5.4. Pyttsx3 Library

Pyttsx3 is a Python text-to-speech library. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application calls the factory function pyttsx3.init () to get a reference to pyttsx3. Engine instance. It is a very easy-to-use tool that converts entered text into speech. The pyttsx3 module supports two female voices and the second is male which is provided by "sapi5" for windows. [25]

2.5.4.1. Supports three TTS engines

- sapi5 - SAPI5 on Windows
- nsss - NSSpeechSynthesizer on Mac OS X
- espeak - eSpeak on every other platform.

2.5.5. Selenium WebDriver

Selenium WebDriver is a web framework that permits you to execute cross-browser tests. This tool is used for automating web-based application testing to verify that it performs expectedly. [26]

Selenium WebDriver allows you to choose a programming language of your choice to create test scripts.

2.6. Hardware Implementation

2.6.1. NIVIDIA Jetson

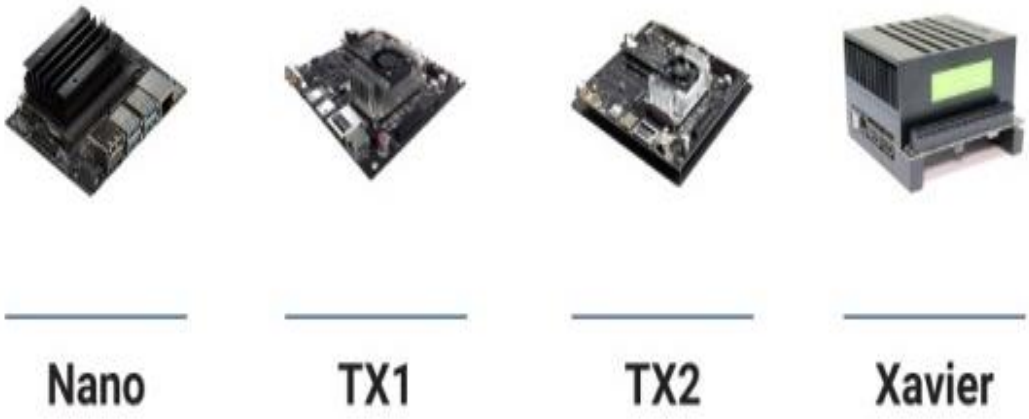

Figure 4: NIVIDIA Jetson Modules

Nvidia Jetson is a series of embedded computing boards from Nvidia. The Jetson TK1, TX1, and TX2 models all carry a Tegra processor (or SoC) from Nvidia that integrates an ARM architecture central processing unit (CPU).[27] Jetson is a low-power system and is designed for accelerating machine learning applications.

In our project, we use the Nvidia Jetson TX2. The Nvidia Jetson TX2 board bears a Tegra X2 of microarchitecture GP10B (SoC type T186 or very similar). This board and the associated development platform were announced in March 2017 as a compact card design for low power scenarios, e.g. for the use in smaller camera drones.

Figure 5: NIVIDIA Jetson TX2 Developer Kit

2.6.2. Software

Various operating systems and software might be able to run on the Jetson board series:

2.6.2.1. Linux

JetPack is a Software Development Kit (SDK) from Nvidia for its Jetson board series. It includes the Linux for Tegra (L4T) operating system and other tools

2.6.2.2. QNX

The QNX operating system is also available for the Jetson platform, though it is not widely announced. There are success reports of installing and running specific QNX packages on certain Nvidia Jetson board variants. Namely, the package qnx-V3Q-23.16.01 that is seemingly in parts based on Nvidia's Vibrante Linux distribution is reported to run on the Jetson TK1 Pro board.

2.7. Related Work

At Mobile World Congress 2019 in Barcelona from 25 – 28 February 2019, BMW Group presented BMW Natural Interaction for the first time. The new system combines the most advanced voice command technology available with expanded gesture control and gaze recognition to enable genuine multimodal operation for the first time. The first BMW Natural Interaction functions will be available in the BMW iNEXT from 2021. Just like in interpersonal dialogue, BMW Natural Interaction allows the driver to use their voice, gestures, and gaze at the same time in various combinations to interact with their vehicle. [28] The preferred mode of operation can be selected intuitively, according to the situation and context. Voice commands, gestures, and the direction of gaze can be reliably detected by the vehicle, combined and the desired operation executed. This free, multimodal interaction is made possible by speech recognition, optimized sensor technology, and context-sensitive analysis of gestures. Driver's entire operating environment. Spoken instructions are registered and processed using Natural Language Understanding.

Figure 6: BMW Gaze & Gesture Mode Interaction

An intelligent learning algorithm, which is constantly being refined, combines and interprets complex information so that the vehicle can respond accordingly. This creates a multimodal interactive experience geared towards the driver's wishes. By combining different modalities, vehicle functions can be initiated in different ways.

The driver decides how they want to interact, based on their personal preferences, habits, or the current situation. So, when the driver is engaged in conversation, they would probably choose gesture and gaze control; when their eyes are on the road, better to rely on speech and gestures.

In this way, for example, car windows or the sunroof can be opened or closed, air vents adjusted or a selection made on the Control Display. If the driver wants to learn more about vehicle functions, they can also point to buttons and ask what they do. With enhanced gesture recognition and the car’s high level of connectivity, the interaction space is no longer confined to the interior. For the first time, occupants will be able to interact with their direct surroundings, such as buildings or parking spaces. Even complex queries can be answered quickly and easily by pointing a finger and issuing a voice command. “What’s this building? How long is that business open? What is this restaurant called?

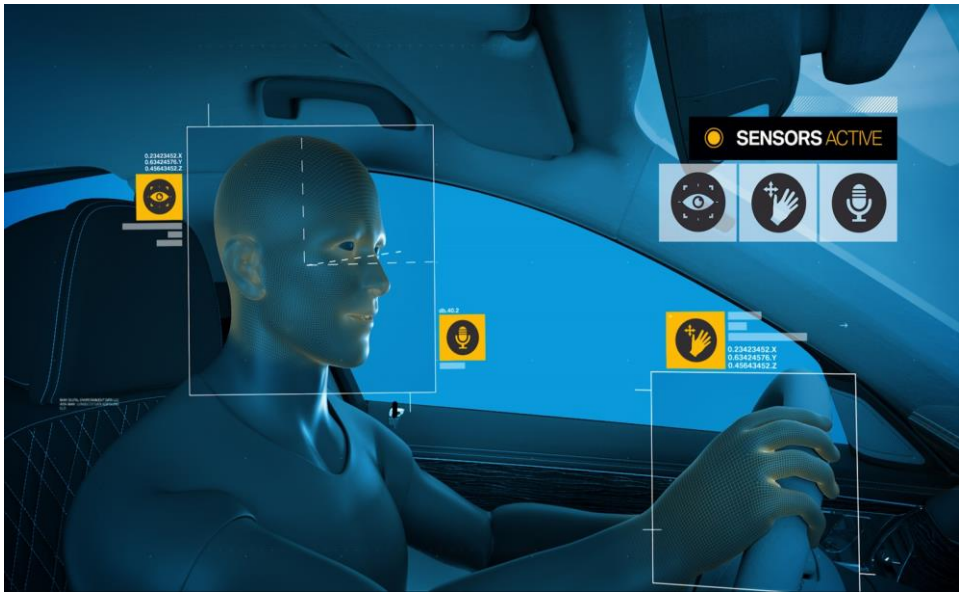

Figure 7: BMW Natural Interaction

2.7.1. BMW Natural Interaction

BMW has always played a pioneering role in the development of systems promoting intuitive operation. In 2001, the BMW Group became the world's first car manufacturer to introduce a new control logic for vehicles, with the iDrive Controller.

The combination of a controller mounted on the center console with a multifunctional Control Display replaced a variety of switches, buttons, and indicators – and is still considered a ground-breaking innovation in the automotive industry. Since 2015, through the use of a 3D camera, BMW gesture control has enabled easy, non-contact operation of various vehicle functions. With the launch of the BMW Operating System 7.0 in 2018, personalized, customizable operation has reached a whole new level, thanks to fully digital displays, optimized speech recognition, and enhanced gesture control. Depending on their personal preferences and situation, the driver can choose between the iDrive controller, steering wheel buttons, touch display or voice, and gesture control. The importance of voice control as the most natural form of interaction is further underlined by the BMW Intelligent Personal Assistant.

2.7.2. Environmental Interaction through Connectivity

Thanks to intelligent networking, the area of BMW Natural Interaction extends beyond the vehicle interior. For example, the driver can point a finger at objects in their field of vision and give related voice commands, such as asking for information about opening hours or customer ratings or reserving a table at a restaurant. Thanks to the vehicle's depth of connectivity, extensive environmental data, and artificial intelligence enable BMW Natural Interaction to transform the vehicle into a well-informed, helpful passenger.

As part of a sophisticated mixed-reality installation, BMW will immerse visitors to Mobile World Congress 2019 in application scenarios where they can experience the customer benefits of BMW Natural Interaction for themselves hands-on. A specially-designed spatial concept and virtual-reality goggles are used to create a thoroughly realistic experience that showcases the new possibilities during a virtual ride in the BMW Vision iNEXT.

Visitors discover the previously unknown freedom of gesture control throughout the area detected by the gesture camera, which extends across the entire width of the front vehicle interior. Initially, in training mode, directional detection of the pointing gesture is visualized by a dynamic light pulse that follows the direction. Objects the driver can interact with via pointing are then highlighted. Just how natural this interaction is becoming apparent in the simple combination of gesture and language.

Figure 8: BMW gesture Controlling

For example, if the driver points to a side window, this is visually highlighted with a frame and the voice command "Open" will then open the chosen window. These new possibilities for interaction with the immediate environment are revealed during an automated journey through a futuristic city the driver is unfamiliar with. The vehicle takes over driving and the visitor embarks on a sightseeing tour of a very different kind – simply pointing at buildings to obtain all the information they need about events and exhibitions. Towards the end of the ride, the user reserves tickets for a cinema they drive by along their route and streams the trailer for the film directly into the vehicle.

Chapter 3.
Face Recognition & Drowsy Detection and Alarming System

Face recognition is a key component of future smart car applications with many uses such as determining whether a person is allowed to operate the vehicle or not. Driver drowsiness is the leading cause of accidents in the world. Due to a lack of sleep and fatigue, drowsiness can occur while driving.

3.1. Face Recognition

3.1.1. Introduction

The challenge is to build a fast and accurate system that can detect, recognize, and verify driver identity. The technology used is a low-cost webcam to take front photos. The system consists of two parts. The first is face detection, which is based on a mixture of classic and fast NN methods. The second is facial recognition and verification, which is based on a combination of Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) techniques. Lighting correction techniques are applied to improve overall performance. The proposed system was tested in a vehicle environment, and its recognition rate was 99.38% with a wrong acceptance rate of 0.62%. The face is detected within 1.5 - 2 seconds.

3.1.2. Face Recognition Modules

Face recognition is performed with OpenCV, Python, and deep learning. Additionally, we made use of Davis King's dlib library and Geitgey's face_recognition module which wraps around dlib's deep metric learning, making facial recognition easier to accomplish. [29]

Our network architecture for face recognition is based on ResNet-34 from the Deep Residual Learning for Image Recognition paper by He et al, but with fewer layers and the number of filters reduced by half.

Keep in mind that we are not training a network here. The network itself was trained by Davis King to create 128-d embeddings on a dataset of ~3 million images. On the Labeled Faces in the Wild (LFW) dataset the network compares to other state-of-the-art methods, reaching 99.38% accuracy.

Both Davis King (the creator of dlib) and Adam Geitgey (the author of the face_recognition module we'll be using shortly) have written detailed articles on how deep learning-based facial recognition works:

- High-Quality Face Recognition with Deep Metric Learning (Davis).
- Modern Face Recognition with Deep Learning (Adam).

The triplet consists of 3 unique face images, 2 of the 3 are the same person and the third image is a *random* face from our dataset and is *not* the same person as the other two images. The NN generates a 128-d vector for each of the 3 face images. For the 2 face images of the same person, we tweak the neural network weights to make the vector closer via distance metric.

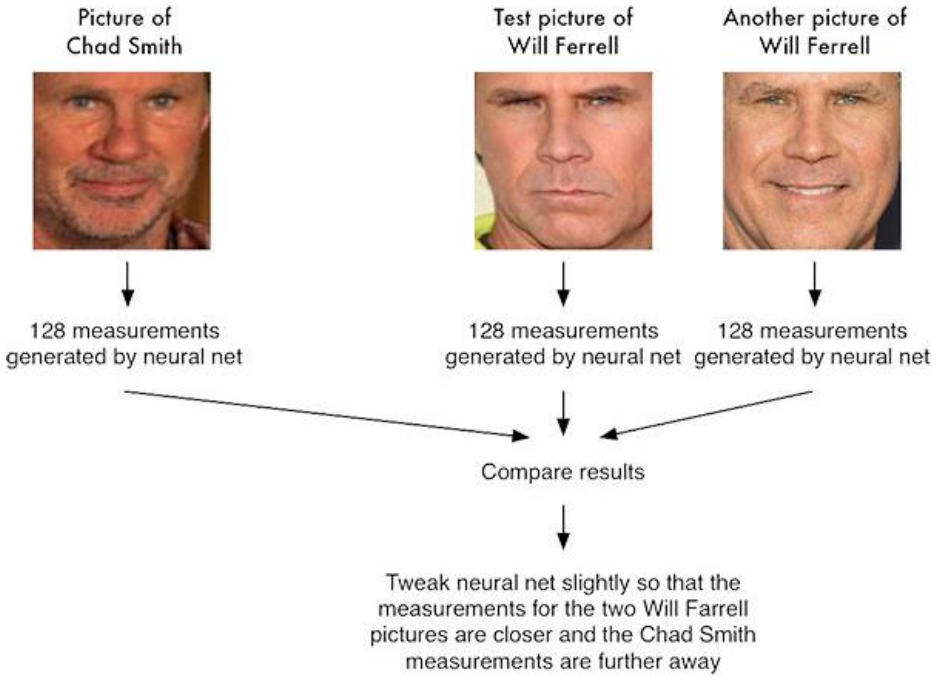

Figure 9: Facial recognition via deep metric learning involves a "triplet training step"

Our network quantifies the faces, constructing the 128-d embedding for each. From there, the general idea is that we'll **tweak the weights** of our neural network so that the 128-d measurements of the two Will Ferrel will be closer to each other and farther from the measurements for Chad Smith.

To perform face recognition with Python and OpenCV we need to install two additional libraries:

- **Dlib**

  The dlib library, maintained by Davis King, contains our implementation of "deep metric learning" which is used to construct our face embeddings used for the actual recognition process.

- **Face_recognition**

  The face recognition library, created by Adam Geitgey, wraps around dlib's facial recognition functionality, making it easier to work with.

3.1.3. Overview of the system

A low-cost webcam on the driver's side dashboard is used to capture the face.[30] During the authentication process, the driver must look straight ahead in front of the camera. The computer is used as a platform for the facial recognition system. The system processes the received data and compares it with the data stored in the template, and in case of recognizing the person as an authorized person, it will send control signals to the input components, which are the latch and ignition keys, and if this person is not recognized, the front image will be taken again, and if the system fails On getting to know this person 3 times, notifications "Unknown Login!!!" will be sent to the car owner telling him that an unknown person is in the car, this message will be sent using an application we developed which needs an internet connection.

⇒ Note that we have applied face recognition to images.

3.2. Drowsy Detection & Alarming System

Driver drowsiness detection is a vehicle safety technology that helps to prevent accidents caused by driver drowsiness. Various studies have suggested that about 20% of all road accidents are linked to fatigue and up to 50% on certain roads.[31] [32]

3.2.1. Introduction

Driver exhaustion is a major variable in a large number of vehicle accidents. Late visions, assessment attributed to 1,200 deaths annually and 76,000 injuries that can be attributed to fatigue-related incidents.

All drowsiness accidents have all the factors that make them more dangerous, due to the high speed involved in distraction and the driver's inability to perform any avoidance activity, or even brake, before the accident occurs.[33]

The best way to avoid accidents caused by drowsiness of drivers is to detect the driver's sleepiness and warn him before bed. To detect drowsiness, several techniques have been used such as retinal detection and facial landmarks. Here in this chapter, we suggest a way to detect eye blinking in video streams using facial landmarks detection and Eye aspect ratio.

3.2.2. The drowsiness detector algorithm

To build our blink detector, we'll be computing a metric called the eye aspect ratio (EAR), introduced by Soukupová and Čech in their 2016 paper, Real-Time Eye Blink Detection Using Facial Landmarks.

We can apply face detection to localize important areas of the face, including the eyes, eyebrows, nose, ears, and mouth. This also means that we can extract specific face structures by knowing the indexes of specific face parts. In terms of flash detection, we are only interested in two sets of facial structures (eyes).

Each eye represented by 6 (x, y) - coordinated, starts from the left corner of the eye (as if you were looking at the person), then works clockwise around the rest of the area.

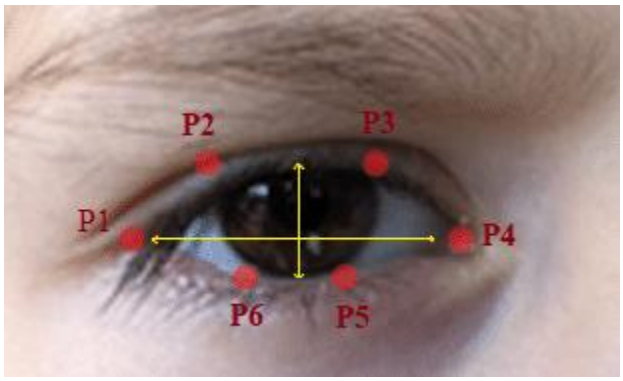

Figure 10: The 6 facial landmarks associated with the eye

Based on this image, we must take a major point, there is a relationship between the width and height of these coordinates.

Based on the work Soukupová and Čech did in the 2016 paper, Real-Time Eye Blink Detection using facial features, we can then derive an equation that reflects this relationship called the EAR:

$$\boldsymbol{EAR} = \frac{\|\boldsymbol{P_2}-\boldsymbol{P_6}\|+\|\boldsymbol{P_3}-\boldsymbol{P_5}\|}{\boldsymbol{2}\|\boldsymbol{P_1}-\boldsymbol{P_4}\|} \quad \rightarrow \quad \textbf{(1)}$$

Where $P_1 \ldots P_6$ are $2D$ facial landmark locations.

The numerator in this equation calculates the distance between the vertical eye circumference while the denominator calculates the distance between the horizontal eye circumference, and the denominator is appropriately distributed as there is only one set of horizontal points but two sets of vertical points.

This equation is interesting as the aspect ratio of the eye is almost constant while the eye is open, but it will quickly drop to about zero when flashing occurs.

Using this simple equation, we can avoid image processing techniques and simply rely on the ratio of historical eye distances to determine whether a person blinking.

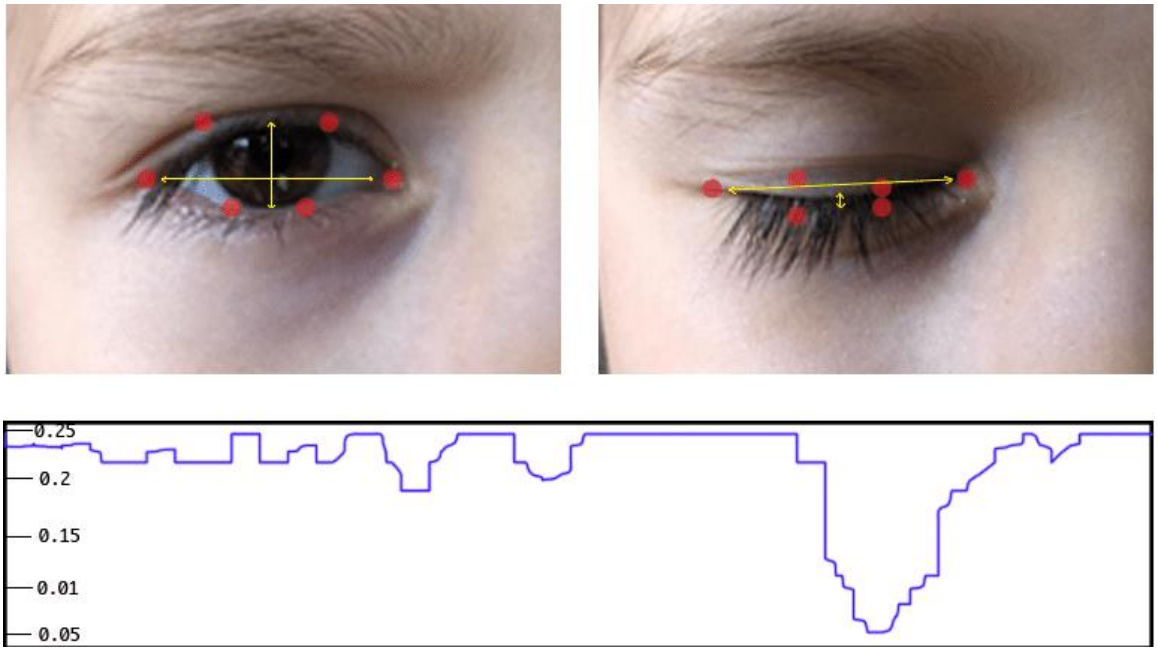

Figure 11: Top-left: A visualization of eye landmarks when then the eye is open. Top-right: Eye landmarks when the eye is closed. Bottom: Plotting the eye aspect ratio over time.

3.2.3. Overview of the system

The general flow of our drowsiness detection algorithm is fairly straightforward. First, we'll set up a camera that monitors a stream for faces. If a face is found, we apply facial landmark detection that is expanded and used to determine the duration of a person's eye closure and extract the eye regions. If their eyes are closed for a certain period, we will assume that they are beginning to fall asleep and start an alarm to wake them up and attract their attention and if this didn't wake him up for another certain period the auto parking system will be activated and a notification will be sent clarifying that the driver is in danger.[34]

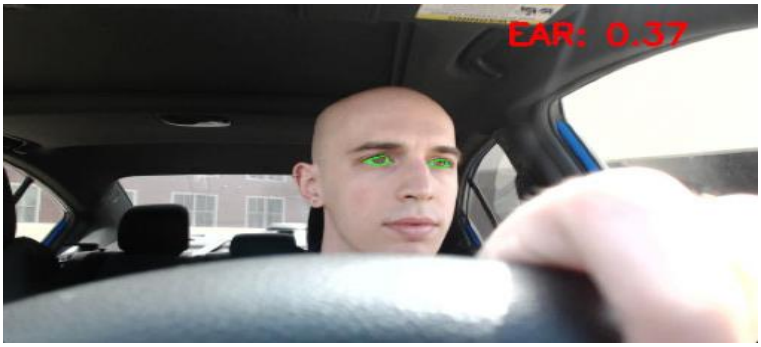

Figure 12: Apply facial landmark localization to extract the eye regions from the face

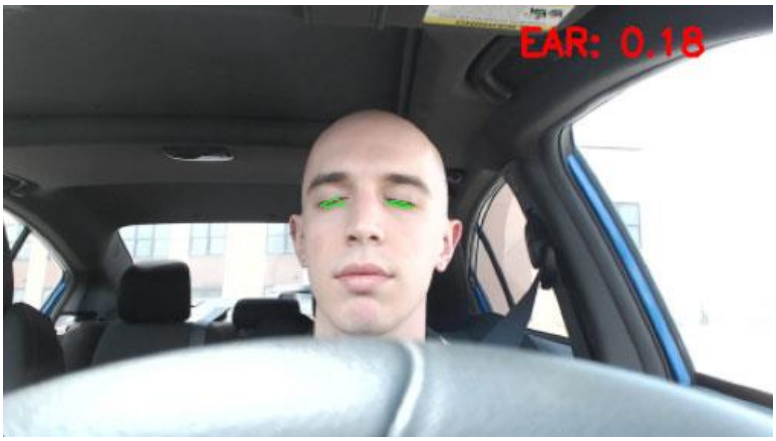

Figure 13: Compute the eye aspect ratio to determine if the eyes are closed

Figure 14: Sound an alarm if the eyes have been closed for a sufficiently long enough time

3.3. Summary

In this chapter, I explained how to create a face recognition and blinking detector using OpenCV, Python, and dlib. The first step in building a blink detector is to perform a facial landmark to locate the eye in a specific frame of the video stream.

Chapter 4.
Voice Recognition

4.1. Introduction

The importance of voice control as the most natural form of interaction has been emphasized by the smart personal assistant. This digital assistant supports the driver in a variety of situations and recognizes his routine and habits with every voice command, making it easy to operate the vehicle and access functions and information by voice, where using your voice in your interactions and commands may be one of the easiest ways if not the easiest way to interact and say the commands without the need of nearly any effort neither any risks of getting the driver's attention away from the road, so it increases the driver's both luxury and safety.

4.2. Model Architecture Construction

To construct a speech commands recognition system let's go through some steps, as of the first step is to specify the words needed for our command to train a model which can recognize them, which in our case there were 20 words which were: {open, close, up, down, increase, decrease, forward, backward, right, left, start, pause, window, first, second, third, fourth, roof, volume, ac}, so to train a model to recognize these words there are two steps, first to collect the data and recordings of these 20 words and second to train the model on these 20 words.

4.2.1. Deep Speech model

Mozilla Deep Speech model was used, giving word error rate of about 7.5%, but after testing it was clarified that it wasn't suitable neither for our accent or for one-word (command) recognition systems.

4.2.2. First Collection of Data

The first trial of collecting the data was that each member of the team was to record each of the 20 words in different conditions, where the data was recorded 10 times with car noises in the background, 10 times with different random noises(Tv, gaussian, people

talking, etc.) in the background, where we are 6 members and 10 times with clear background (noise-free), the team consisted of 6 members each one recording 30 records of each word of the 20 words so we had a total of about 3600 records of these 20 commands under the different condition to train our model on them.

4.2.3. First Trained Model

After collecting the data it was time to start constructing a model to be trained on these dataset but before even constructing the model the data must go through the preprocessing stage, where number of 360 recordings were chosen as a reference where each word of the 20 was taken 9 times (6 of each condition of the 3 conditions they were recorded under), now there are 360 recordings used as references and the rest of the data is about 3420 ,now before starting the training phase the preprocessing will be as following each of the recordings of the dataset will have its features extracted using the MFCC technique where 20 features were collected from each record with sampling rate of 22050 Hz then it will be compared using the DTW technique with each of the features of the 360 references so each record will now have an array of 360 values which are the DTW values between it and each of the reference recordings and then these arrays will be passed to the neural network to train on them, but this method didn't give satisfying results, then the reference was changed into 180,120 and only 20 recordings but all of these gave unsatisfying results.

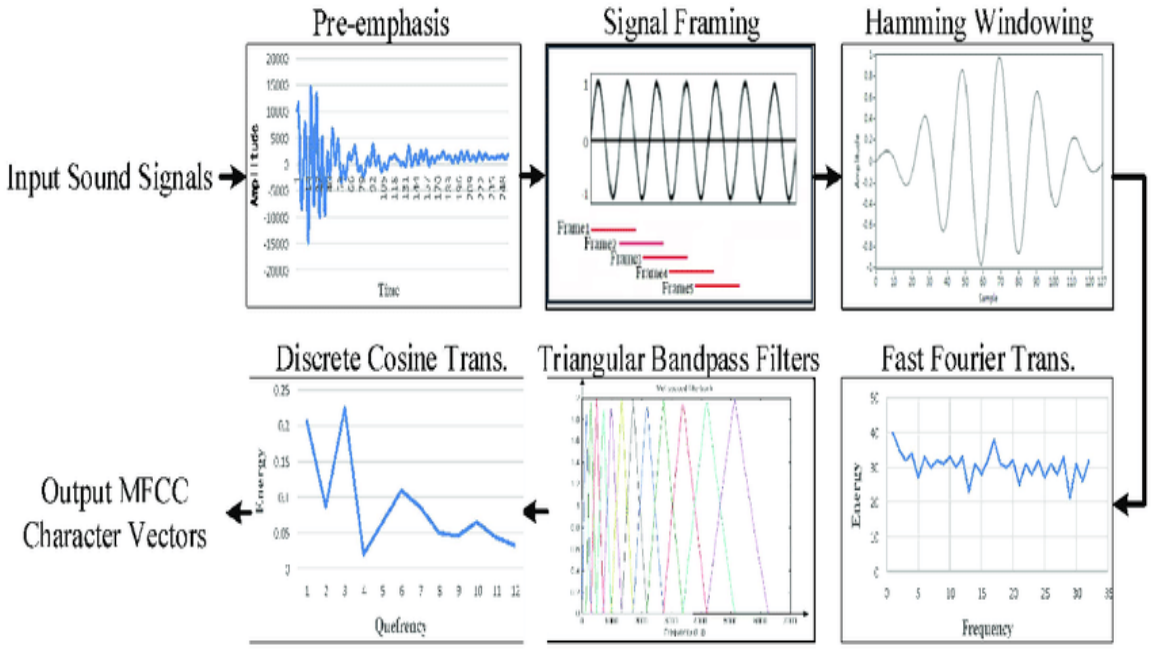

Figure 15: MFCC process

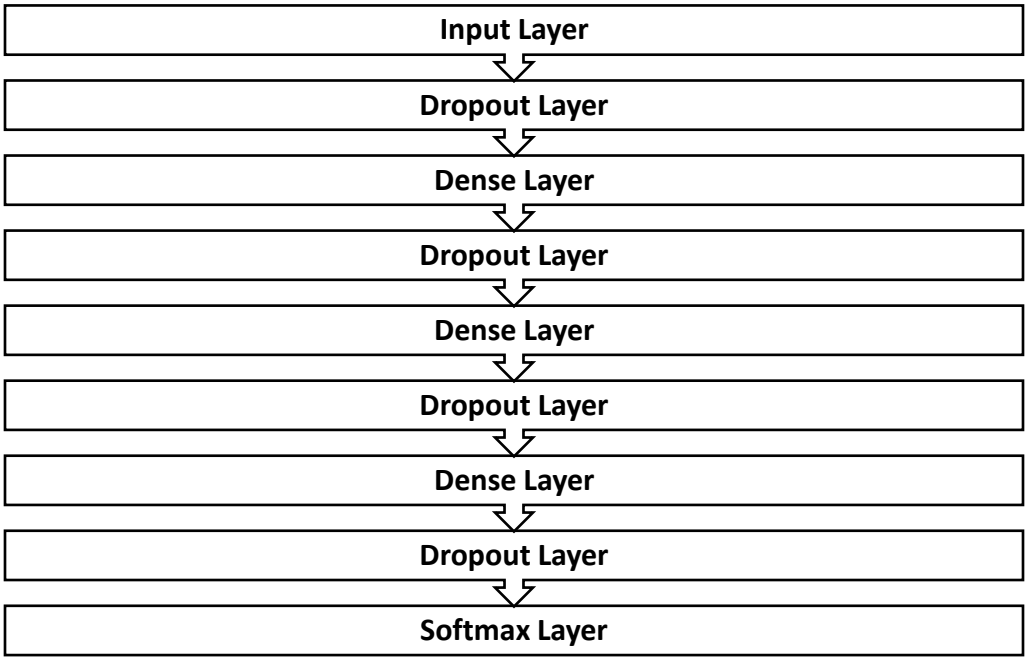

Figure 16: Neural Network Block Diagram

4.2.4. Second Trained Model

It was concluded that the last method caused the network to be as if it was like f memorizing the values of the DTW of each record and it was forcing it to use this method in comparing the features using the DTW and not letting it learn by itself so instead of feeding the network with the array of DTW let's try feeding it with MFCC vector itself which depending on the length of the record was 20xL where the number of features taken at each instant was 20 features and L varied from a record to another so it was fixed on the longest record and any smaller record was zero-padded so that the size will be fixed then this vector was flattened as we used dense layers not convolutional layers, using this method we started seeing the difference in the results but we suffered from overfitting.

4.2.5. Second Collection of Data

To solve the overfitting problem that occurred it was required to increase the collected data so another phase of collecting the data from random people under random and various cases and conditions was done so that the number of the recordings was about 9000 records which were more than double the dataset collected before.

4.2.6. Third Trained Model

Now let's try training our model using the same method but using the new dataset which is by far greater than the last time, after the training phase the results were greater than before but it was still not enough and still suffered from some overfitting where the validation accuracy was too stuck at about 70-80% and the training accuracy was to exceed 96%.

4.2.7. Fourth Trained Model

After the last results it was clear that using this dataset wasn't enough to train the model on all the cases and all the types of noise ,so this time it will be tried to do start-end point detection for each record before training and also clearing the noise which were done using the sox and ffmpeg libraries, where each record will only contain the command required without any silence or noise in it, training the model using the optimized data gave magnificent results using both the dense layers constructed before and also when tried on convolutional layers and an accuracy about 95% was reached this time and the model had a performance of about 4-5 seconds working on a 2^{nd} generation CPU which of course will be better if used on better capable GPU, but this model couldn't perform well under all kinds of noise and all sounds where of course due to training with no noise and most of the records being of only the 6 team members caused it to be overfitted on some cases but it was a good model as a proof of concept.

4.2.8. Final Model

The last model gave good results and accuracy but only as a proof of concept model but it wasn't capable as it couldn't work under all the conditions, so Google Speech-to-Text API was used due to its high accuracy and capability under different conditions as it can convert audio to text by applying powerful neural network models in an easy-to-use API. The API recognizes more than 120 languages and variants.

Chapter 5.
Intelligent Personal Assistant

Drivers and passengers will be able to use the "Wake up Assistant" command to activate the system. Drivers will be able to connect with their cars and access their functions and information simply by speaking.

5.1. Introduction

Intelligent Personal Assistant is the automated equivalent of Amazon's Alexa or Google Home, while each assistant may specialize in slightly different tasks, they don't have to search for a keyboard to find answers to questions like "What is the weather today?" Or "Where is Switzerland?". Where each seeks to facilitate the user's life through verbal interactions. Artificial intelligence that relies on the cloud voice command system that enables you to do things like moving the window or changing the cabin temperature without raising your hands off the wheel.

In short, the intelligent personal assistant is the perfect auxiliary driver and comes especially useful during daily driving ("Wake up Assistant, find the nearest gas station on our way"). But Intelligent Personal Assistant is an entertainment expert as well. So, he can immediately search for stations about the type of music required ("Play Classic Music Please") and search for any information you want ("Search for Barack Obama") and return the information to speak to you.

The Intelligent Personal Assistant will also be able to get information for the driver from outside his vehicle, and it can also provide information about the vehicle, from questions about how to operate the systems or the condition of the vehicle.

5.2. Building a Digital Assistant in Python

This section will guide you in the basics of building a smart digital personal assistant in Python, with voice activation as well as responding. From there, we can customize it to perform any tasks we need.

5.2.1. Speech Recognition Package

When you ask a question, we will need something to take. The Speech Recognition Package allows Python to access audio from your device's microphone, transfer audio, save the audio to an audio file, and other similar tasks. The SpeechRecognition library is used to activate the device's microphone, then convert the audio into text in a string. We find it reassuring to print a phrase when the microphone is activated, in addition to the text above the microphone hears, so we know it is working properly. We also include conditions to cover common errors that may occur if there is a lot of background noise, or if the Google Cloud Speech API request fails.

5.2.2. Text-to-speech package

Our assistant will need to convert your question to text. After that, once the assistant searches for an answer online, it will need to convert the response into an audible phrase. For this purpose, we will use pyttsx3 that is a Python text-to-speech library. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3.

The pyttsx3 module supports two female voices and the second is male which is provided by "sapi5" for windows. It is a very easy-to-use tool that converts entered text into speech.

5.2.3. Wikipedia Package

Wikipedia, the largest free encyclopedia in the world. It is a land full of information. I mean who would have used Wikipedia all their life (if you haven't used it, then you are most likely lying). The python library called Wikipedia allows us to easily access and analyze data from Wikipedia. In other words, you can also use this library as a small scraper as you can only scrape limited information from Wikipedia.

Wikipedia-API is an easy-to-use Python wrapper for Wikipedia’s API. It supports extracting texts, sections, links, categories, translations, etc. The goal of Wikipedia-API is to provide a simple and easy-to-use API for retrieving information from Wikipedia. For an essay summary, The WikipediaPage chapter contains a Property Summary, which displays a description of the Wiki page.

Also, the Wikipedia API provides us with an option to change the language in which we want to read articles. All you have to do is set the language to the one you want. Wikipedia currently supports 444 different languages.

5.2.4. Selenium WebDriver Package

Selenium WebDriver is a collection of open source APIs used to automate web application testing. This tool is used to automate the testing of a web application to verify that it works as expected. It supports many browsers like Safari, Firefox, IE, and Chrome. We use Chrome so we installed the geckodriver.

5.3. Overview of the system

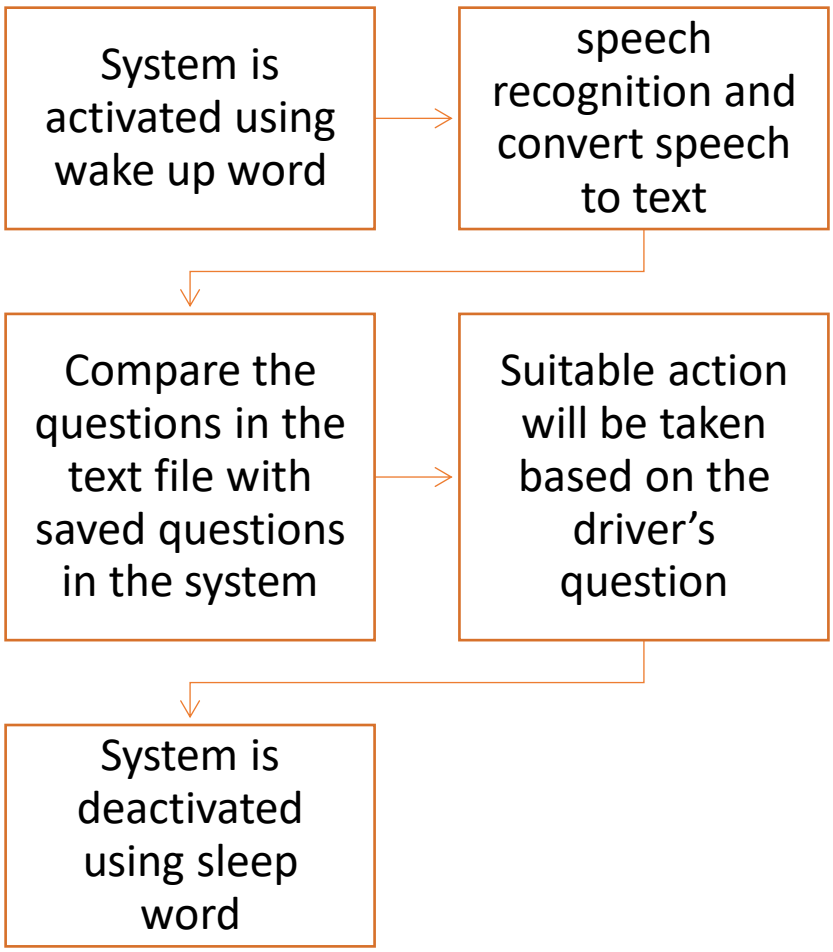

Figure 17: System overview

Our system starts working as the driver starts using the vehicle, it periodically checks if the wake up word has been said correctly to activate the assistant, after activation the assistant asks for the voice question and activate the recorder for 5 second to save the voice, in case of any error occurs due to noise in the environment the assistant reply with message “Could not understand your audio, Please try again”, then after saving the audio successfully the speech recognition process starts and converts it to text, this text is processed to extract the driver question and take the suitable action, finally the system checks if the sleep word has been said to deactivate the assistant.

We ask a lot about the places near my site. This usually means that I open a new tab in the browser and search for it on Google Maps. Of course, if my new digital assistant can do it for me, it will save me. To implement this feature, we will add a new function to our digital assistant. The new function will be identified in the case of "What is this building?" In the voice search query and append the following word to the Google Maps URL, the assistant responds and issues the OS to open Chrome with the specified URL. My Location will open in Chrome, displaying latitude and longitude, and the Google API uses Latitude and Longitude to get information about the nearby location you inquired about.

Figure 18: Intelligent Personal Assistant

5.4. Summary

Through speech recognition, improved sensor technology, and context-sensitive gesture analysis, multimedia interactivity is possible. Spoken instructions are recorded and processed using natural language understanding. The constantly revised smart learning algorithm integrates and interprets complex information so that the vehicle can respond accordingly. This creates an interactive multimedia experience geared towards the driver's desires. We are using Google speech recognition API and google text to speech for voice input and output respectively. The implemented assistant can open up the application (if it’s installed in the system), search Google, Wikipedia, and YouTube about the query, etc. by just giving the voice command. We can process the data as per the need or can add functionality.

Chapter 6.
Gesture Recognition

6.1. Introduction

Gestures are an important aspect of human interaction, both interpersonally and in the context of man-machine interfaces. There are many faces to the modeling and recognition of human gesture, gestures can be expressed through hands, faces, or the entire body. Gesture recognition is an important skill for machines which human interacts with since gestures help to clarify spoken commands and are a compact means of communicating geometric information. It provides a redundant form of communication between the user and the machine. When command and gesture are used together, the machine needs only recognize one of two commands, which is crucial in situations where speech may be garbled or drowned out (e.g., in space, underwater, on the battlefield). Gestures are also an easy way to give geometric information to the machine/Vehicle. Rather than give coordinates to where the machine should move or use the command on, the user can simply point to an object or a certain place.

6.2. The Implementation Approach

In this section, we will discuss the different models which we used to implement the Gesture direction Detection subsystem.

6.2.1. Hand tracking using Bounding box detection model

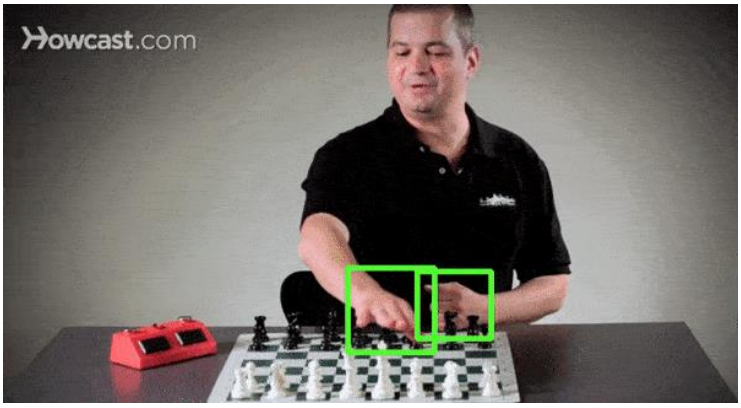

Figure 19: Hand Detection using Neural Networks (SSD) on TensorFlow

As the first step in this block, we used a bounding box method to track the hand of the driver as shown in Figure18, it simply tracks the hand movement and surrounds it with a box. The deep learning model which has been used is SSD model.

SSD model has a significant improvement in speed for high-accuracy detection than other detection models. [35]

The main idea of SSD is predicting scores of the category and box offsets for a fixed set of default or reference bounding boxes using small convolutional filters applied to feature maps, also the model produces predictions in different scales from different scales feature maps, and separate predictions by aspect ratio to provide high accuracy detection.

The SSD model as shown in Figure 19 is based on a feed-forward convolutional network which based on a standard architecture used for image classification followed by a non-maximum suppression step, the first feed-forward convolutional network produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in the boxes and non-maximum suppression step is used to produce the final detections.

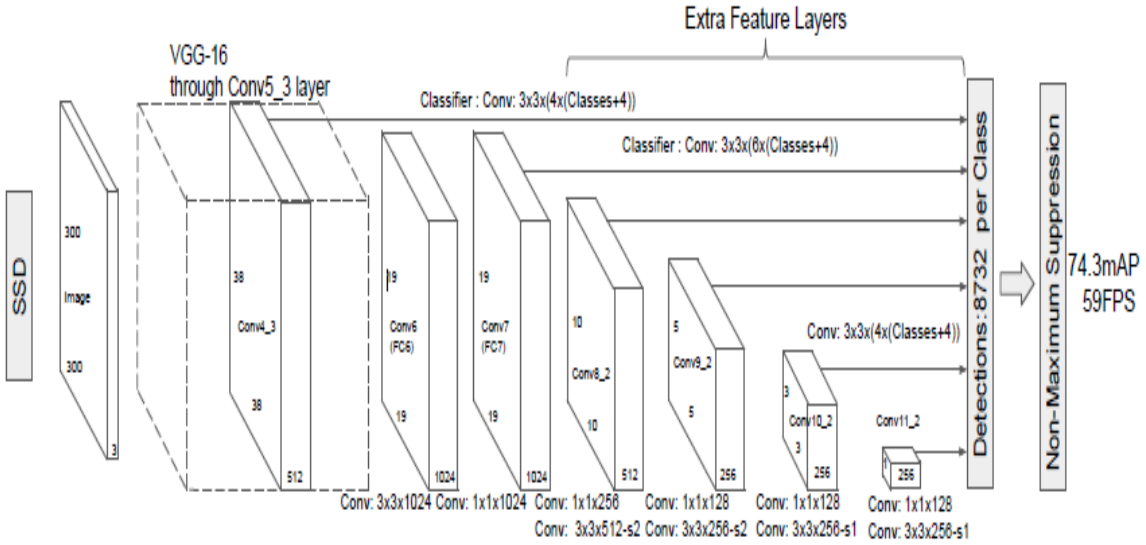

Figure 20: The Architecture of Single Shot MultiBox Detector (SSD) Model

The key features which model used to produce the detection are:

Multi-scale feature maps for detection

SSD model adds some feature layers to the end of the network which decreases in size progressively and predicts the offsets to default boxes of different scales and aspect ratios and the confidences. The convolutional model for predicting detections is different for each feature.

Convolutional predictors for detection

Any feature layer in the added layers or optionally in the base network uses a set of convolutional filters to produce a fixed set of detection predictions.

Default boxes and aspect ratios

The model associates a set of default bounding boxes with each feature map cell, for multiple feature maps at the top of the network and each box has a fixed position relative to its corresponding cell because of the default boxes tile the feature map in a convolutional manner. Apply these default boxes to several feature maps of different scales discretize the space of possible output box shapes.

Table 1: Advantages and Disadvantages of SSD Model

Performance	25 FPS
Accuracy	~98%
Advantage	1. High performance. 2. High Accuracy. 3. Easier Code implementation.
Disadvantage	Unbeneficial in angle determination

⇒ Note that the performance is measured as the time which the model takes to process one frame, so whenever this time is short the model will be fast and more suitable for real-time applications, also can be is measured as the number of frames which had been processed in one second (unit FPS).

6.2.2. Key Point Detection: Multi-view bootstrapping

While many approaches to image-based face and body key-point localization exist, there are no marker-less hand key-point detectors that work on RGB images in the wild. This method enables real-time 2D hand tracking in single view video and 3D hand motion capture. Unlike the face and body, large datasets of annotated key-points do not exist for hands. Generating such datasets presents a major challenge compared to the face or body. Due to heavy occlusions, even manual key-point annotations are difficult to get right. For the key-points that are occluded, the annotated locations are at best an educated guess. This method is called Multiview bootstrapping which is based on the following observation, if a particular image of the hand has significant occlusion, there often exists an un-occluded view.

Multiview bootstrapping systematizes this insight to produce a more powerful hand detector which is demonstrated. It allows a weak detector, trained on a small annotated dataset, to localize subsets of key-points in good views and uses robust 3D triangulation to filter out incorrect detections. Images, where severe occlusions exist, are then labeled by reprojections of the triangulated 3D hand joints. By including these newly generated annotations in the training set, we iteratively improve the detector, obtaining more and more accurate detections at each iteration. This approach generates geometrically consistent hand key-point annotations using Multiview constraints as an external source of supervision. In this way, we can label images that are difficult or impossible to annotate due to occlusion.

- **Training:**

Table2 describes the overall procedure for Multiview bootstrapping, by using $\{I_v{}^f : v \in [1 \dots V], f \in [1 \dots F]\}$ as an input set of unlabeled Multiview image frames, with v iterating over the V camera views, and f iterating over F distinct frames (i.e., time instants, so one frame represents V images).

Table 2: Overall Procedure for Multiview Bootstrapping

Algorithm 1	
Inputs	Unlabeled images: $\{\{I_v{}^f : v \in \text{views}, f \in \text{frames}\}$
	Key-point detector: $d_o(I) \rightarrow \{(x_p, c_p) for\ p \in points\}$
	Labeled training data: τ_o For iteration I in 0 to K: 1. Triangulate key-points from weak detections For every frame f: a. Run detector $d_i(I_v{}^f) on\ all\ views\ v$ b. Robustly triangulate key-points. 2. Score and sort triangulated frames 3. Retrain with N-best reprojections $d_{i+1} \leftarrow train(\tau_o \cup \tau_{i+1})$
Outputs	Improved detector $d_K(.) and\ training\ set\ \tau_K$

- **Convolutional Pose Machine (CPMs):**

For the detectors d_i, the architecture of Convolutional Pose Machines (CPMs) [36], with some modification. CPMs predict a confidence map for each key-point, representing the key-point location as a Gaussian centered at the true position. The predicted confidence map corresponds to the size of the input image patch, and the final position for each key-point is obtained by finding the maximum peak in each confidence map.

Figure 20 describes the CPM architecture [36], it shows a convolutional architecture and receptive fields across layers for a CPM with any T stages. The pose machine [37] is shown in insets (a) and (b), and the corresponding convolutional networks are shown in insets (c) and (d). Insets (a) and (c) show the architecture that operates only on image evidence in the first stage. Insets (b) and (d) shows the architecture for subsequent stages, which operate both on image evidence as well as belief maps from preceding stages. The architectures in (b) and (d) are repeated for all subsequent stages (2 to T). The network is locally supervised after each stage using an intermediate loss layer that prevents

vanishing gradients during training. Below in inset (e), we show the effective receptive field on an image (centered at the left knee) of the architecture, where the large receptive field enables the model to capture long-range spatial dependencies such as those between head and knees.

It is proposed to use the convolutional stages of a pre-initialized VGG-19 network up to conv4_4 as a feature extractor, with two additional convolutions producing 128-channel features F. For an input image patch of size $w \times h$, the resulting size of the feature map F is $w' \times h' \times 128$, with $w' = \frac{w}{8}$ and $h' = \frac{h}{8}$. There are no additional pooling or downsampling stages, so the final stride of the network is also 8.

This feature map extraction is followed by a prediction stage that produces a set of P confidence or score map $S^1 = \{{S_1}^1 \dots {S^1}_P\}$, one score map ${S_P}^1 \epsilon R^{w' \times h'}$ for each key-point p. Each stage after the first takes as input the score maps from the previous stage S^{t-1}, concatenated with the image features F, and produces P new score maps S^t, one for each key-point. We use 6 sequential prediction stages, taking the output at the final stage S^6. We resize these maps to the original patch size ($w \times h$) using bicubic resampling and extract each key-point location as the pixel with maximum confidence in its respective map. [38] [39] [40]

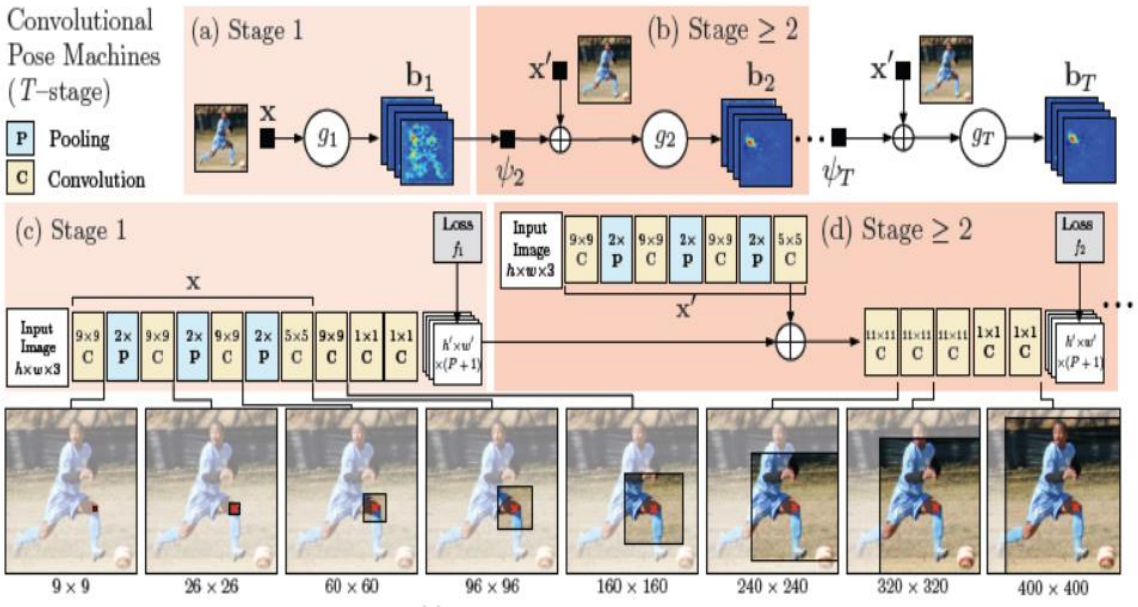

Figure 21: Convolutional Pose Machines architecture

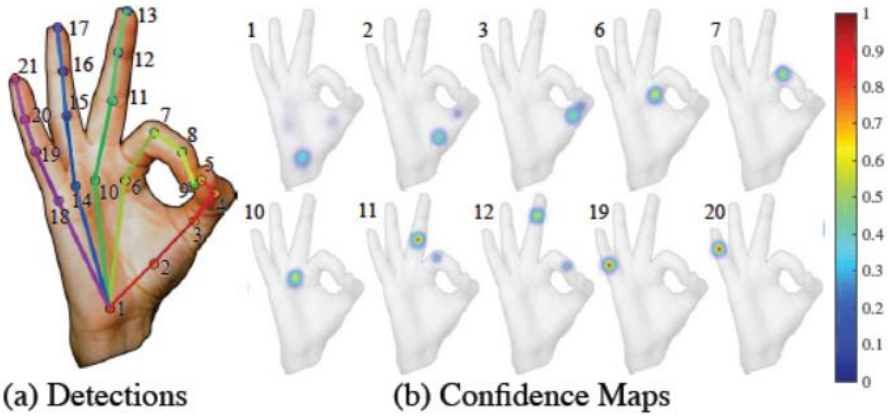

Figure 22: (a) Input image with 21 detected key-points. (b) Selected confidence maps produced by the proposed detector

To get rough estimates of the hand key-points, a huge multi-view system is set up to take images from different viewpoints or angles consists of 31 HD cameras. This model produces 22 key-points where the hand has 21 points and the 22nd point signifies the background.

In our project, the proposed gesture model is used and tested with 2 different frameworks to get the best results from both methods. One method is to use the Caffe framework and the second method is the TensorFlow implementation of convolutional pose machine. [41]

6.2.2.1. Caffe Framework Implementation

Caffe is a deep learning framework developed by the Berkeley Vision and Learning Center (BVLC). The model is trained by executing one Caffe command from the terminal. After training the model, we will get the trained model in a file with the extension ".caffemodel". The ".caffemodel" trained model is used to make predictions of new unseen data. [42]

The Caffe model weights are used with 'OpenCV' where the output has 22 matrices with each matrix being the Probability Map of a key-point. For finding the exact key-points, the probability map is scaled to the size of the original image. Then it finds the location of the key-point by finding the maxima of the probability map. This is done using the minmaxLoc function in OpenCV. We draw the detected points along with the numbering on the image.

6.2.2.2. CPM TensorFlow Implementation

It is an implementation of the Caffe model but is used with TensorFlow, with the addition of some features such as the Kalman filter to smooth the pose estimation.

Table 3 describes the advantage and disadvantages of both implementations.

Table 3: Caffe model and model Version comparison in Key Point Detection

Implementation	Caffe Version	TensorFlow Version
Performance	Approx. 0.4 FPS	Approx. 4 FPS
Advantage	1. Easy direction determination 2. High Accuracy 3. Easier Code implementation	1. Easy direction determination 2. High accuracy
Disadvantage	1. Very Bad Performance 2. Unbeneficial in angle determination	1. Bad Performance 2. Unbeneficial in angle determination
Issues	• OpenCV gemm function used in convolution is weak and it is very slow with the CPM model. • Pipelining makes sizing errors.	• Slow frame processing due to the large model used. • Key-points are sometimes centralized, due to error in the frozen graph used for hand tracking.

6.2.3. Kinect V2 Joint Detection and Tracking Models

The main problem in the previous approaches was the insufficiency of pointing direction determination and only focusing on the hand gesture recognition or tracking, so it is decided to try a different approach which concentrates on tracking the driver arm not only his hand, then detects the position of the arm joints (I.e. Shoulder, elbow and wrist) and finally calculate the angle with these points using simple Trigonometric Functions

6.2.3.1. Kinect Device Models

Kinect device which is a motion-sensing input device as mentioned in chapter 2 has a great feature called Skeleton tracking that provides very accurate tracking of the human body skeleton and determines the position of all body joints with high performance up to 30 FPS as shown in figure 22.

In the following section, we develop several models using the Kinect device in different software environments to assert the functionally of this approach using different Kinect software libraries depends on the nature of the operating environment.

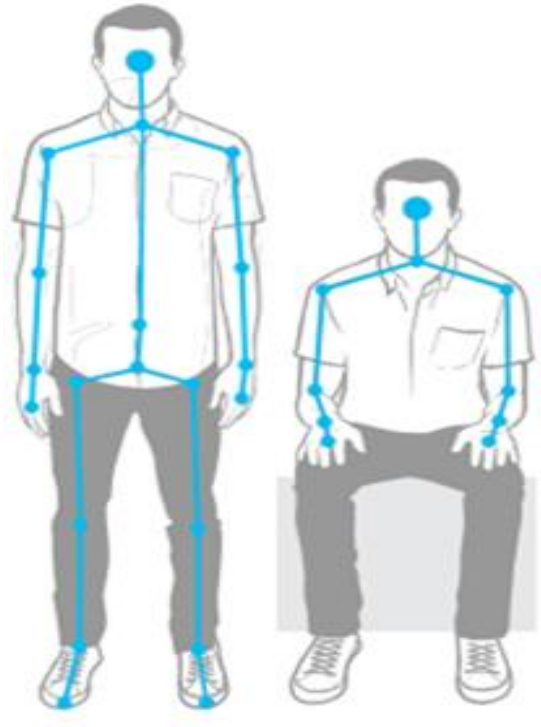

Figure 23: Skeleton Tracking Feature in Kinect

6.2.3.2. Microsoft Kinect SDK model

As Microsoft produces Kinect, it releases an SDK which enables developers to create gesture and voice recognition applications on computers that run the Windows operating system.

6.2.3.3. Kinect with Linux

Kinect v2 got good support on Windows with Kinect SDK but no good support on Linux. Rovers are working at the Italian Society of Mars and Space Suits at the Austrian Space Forum already on Ubuntu 14.04, and it is recommended that the project be implemented on a Linux machine. Linux has up to now stable and standard support for Kinect v2 but there are two options available. Below are the libraries that can be considered for the Kinect v2 hack.

1. Libfreenect2 from OpenKinect

2. Openni2

3. PyKinect

After checking all the libraries, libfreenect2 gave good results and it will be usable with Linux to keep it working. It can be used generally with C ++ but Pylibfreenect2 provides a Python interface for that. The ability of Kinect to track the hand helped us to implement gestures and get angle among shoulder, elbow and wrist using skeleton joints [43]. The ability of the software in combination with the hardware can easily identify the human movements, and this helped us to create Touch-fewer interfaces [44]. The user can follow the gestures independent of their language and control the car.

Figure 24: Angle Calculations with Kinect

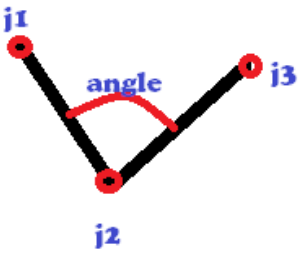

Figure 25: Angle among three joints

The angle is a combination of two lines. To measure the angle, all you need is 3 points in the 3D or 2D space: a starting point, a midpoint and an endpoint. Kinect's power is the ability to calculate the locations of 25 joints in the human body.

The joint is a structure that includes:

- The position in the three-dimensional space.
- Type/name of the joint.
- Tracking accuracy.

The position of the joint in 3D is expressed as CameraSpacePoint. To properly position the 3D position in the 2D position, you need to use Coordinate Mapping.

To measure your elbow angle. This angle consists of 3 joints (shoulder, elbow, and wrist). The middle point of the angle is of course the elbow. The starting point of the angle is the shoulder. The endpoint of the angle is the wrist. Mathematical calculations are extremely helpful when you need to compare the relative positions between a few points. Gesture detection, sign language identification, or even facial expressions. [45]

$$Dx_{12} = X_1 - X_2$$

$$Dy_{12} = Y_1 - Y_2$$

$$Dz_{12} = Z_1 - Z_2$$

$$\boldsymbol{Dnorm_{12}} = \sqrt{\boldsymbol{Dx_{12}}^2 + \boldsymbol{Dy_{12}}^2 + \boldsymbol{Dz_{12}}^2} \rightarrow \qquad \textbf{(2)}$$

$$Dx_{32} = X_3 - X_2$$

$$Dy_{32} = Y_3 - Y_2$$

$$Dz_{32} = Z_3 - Z_2$$

$$\boldsymbol{Dnorm_{32}} = \sqrt{\boldsymbol{Dx_{32}}^2 + \boldsymbol{Dy_{32}}^2 + \boldsymbol{Dz_{32}}^2} \rightarrow \quad \textbf{(3)}$$

$$P = \underline{L}_{12} \,.\, \underline{L}_{32} = Dx_{12} * Dx_{32} + Dy_{12} * Dy_{32} + Dz_{12} * Dz_{32}$$

$$\boldsymbol{Norm\ value} = \frac{\boldsymbol{P}}{\boldsymbol{Dnorm_{12} * Dnorm_{32}}} \quad \rightarrow \quad \textbf{(4)}$$

$$where\ Norm\ value\ in\ \{-1,1\}$$

$$\boldsymbol{angle}_{\ rad} = \boldsymbol{cos^{-1}(Norm\ value)} \quad \rightarrow \quad \textbf{(5)}$$

$$\boldsymbol{angle}_{\ degree} = \boldsymbol{angle}_{\ rad} * \frac{\boldsymbol{180}}{\boldsymbol{\pi}} \quad \rightarrow \quad \textbf{(6)}$$

Where

$\boldsymbol{Dx_{nm}}$**:** The difference between x values of joint n and joint m.

$\boldsymbol{Dy_{nm}}$: The difference between y values of joint n and joint m.

$\boldsymbol{Dz_{nm}}$: The difference between z values of joint n and joint m.

$\boldsymbol{Dnorm_{nm}}$: The normalized difference between joint n and joint m.

$\boldsymbol{P}$**:** The dot product between $\underline{L}_{12}\ and\ \underline{L}_{32}$.

$\underline{\boldsymbol{L}}_{\boldsymbol{nm}}$: The vector between joint n and joint m

$\boldsymbol{angle}_{\ rad}$: Angle in radian

$\boldsymbol{angle}_{\ degree}$**:** Angle in degree

6.2.4. Human Pose Estimation in OpenCV using OpenPose MobileNet

The problem of estimating the human condition can be defined as computer vision techniques that predict the location of different human major points (joints and features) such as elbows, knees, neck, shoulder, hips, chest, etc.

It is a very difficult problem due to various factors such as small, hard to see parts, blockages, and major joint changes. The figure below illustrates the challenges:

Figure 26: Human Pose Estimation Challenge

6.2.4.1. Pose Estimation

OpenPose is a library to discover real-time multi-person key points and multiple interfaces written in C ++ with Python shell available. OpenPose won the key point challenge at Coco 2016. OpenCV has integrated OpenPose into the new Deep Neural Network (DNN) unit. This means that you will need OpenCV version 3.4.1 or higher to run the Estimate Code.

This convolutional neural network-based approach attacks the problem using a multi-stage classifier where each stage improves the results of the previous one. [46]

6.2.4.2. Key-points Detection Datasets

Until recently, little progress had been made in assessing the situation due to the lack of high-quality data sets. This is the enthusiasm in artificial intelligence these days that people think every problem is just a good dataset out of demolition. Some difficult data sets have been released in the past few years, making it easier for researchers to attack the problem with all their intellectual strength.

Some of the data sets are:

1. COCO Key-points Challenge.
2. Datagram of human form MPII.
3. VGG Pose dataset.

6.2.4.3. Pre-trained Models

Figure 27: COCO Key-points vs MPII Key-points

One model was trained in a multi-person data set (MPII) and the other was trained on a COCO dataset. The COCO model produces 18 points, while the MPII model produces 15 points. The outputs drawn on a person are shown in the previous figure.

MobileNet

Based on the MobileNet paper [47], 12 convolutional layers are used as feature-extraction layers. To improve on a small person, minor modification on the architecture have been made.

Three models were learned according to network size parameters.

- **Mobilenet**
  368x368: checkpoint weight download
- **mobilenet_fast**
- **mobilenet_accurate**

TensorFlow Graph File (pb file)

Keras saves its weight in a hierarchical data format (.hdf5) file which slows the loading of the model on Nvidia Jetson TX2, we have created an optimized (converted it to a frozen graph base model; here the value of all variables are embedded in the graph itself thus the protocol buffers (.pb) file cannot be retrained) frozen file of our Keras model based on Tensorflow. Keras does not include by itself any means to export a Tensorflow graph as a (. pb) file, but we could do it using regular Tensorflow utilities. [48]

Before running the demo, you should download graph files. You can deploy this graph on your mobile or other platforms.

- **cmu (trained in 656x368).**
- **mobilenet_thin (trained in 432x368).**

6.2.4.4 Make Predictions and Parse Key-points

Once the image is passed to the model, predictions can be made using a single line of code. The **forward** method for the DNN class in OpenCV makes a forward pass through the network and is another way of saying it is predicting.

We check whether each key-point is present in the image or not. We obtain the location of the key-point by finding the maxima of the confidence map of that key-point. We also use a threshold to reduce false detections. Since we know the indices of the points before-hand, we can draw the skeleton when we have the key-points by just joining the pairs. [49]

Chapter 7.
Natural Interaction

Computing aims to provide an effective and natural interaction between humans and computers. The important goal is to enable computers to understand the situations people have expressed so that personal responses can be provided accordingly. In this chapter, we explore the interaction between the driver and his car from continuous speech and suggest a real-time speech recognition system. The system consists of voice activity detection, speech splitting, feature extraction, gesture recognition, and gaze discovery.

7.1. Functionality

The natural interaction technology allows the driver to combine sound, gesture, and gazing in various ways to interact with their car. The driver decides how he wants to interact with the car using either sound or gestures, based on their personal preferences or current situation.

The system combines the most advanced voice command technology available with expanded gesture control and gesture recognition to enable multimedia operation.

7.1.1. Internal Interaction

7.1.1.1. Face Recognition Interaction

Facial recognition can interact with the car as the system login information where a picture was taken through the car's webcam and compared to other images in the database if the driver's face is known to the system, now the driver can enter the car system and control a car.

But if the driver’s face is unknown to the system, the system will allow another opportunity to uncover the driver’s face if the system fails 3 times in a row to recognize the driver’s face. The system will remain closed and a notification will be sent to the owner's Android phone "Unknown login!!!" telling him that an unknown person is in the car.

7.1.1.2. Gaze Tracking Interaction

Detect drowsiness by tracking the ratio of the eye, if it drops to less than 0.3 for more than 3 seconds, the system will start alerting to try to wake the driver but if it does not wake up and open its eyes for more than 6 seconds after the alarm starts.

The system will switch to automatic parking mode and a notification message that uses the same application in the facial recognition block will be sent to the trusted person as the driver is at risk.

7.1.1.3. Gesture Recognition & Voice Commands Interaction

By accurately detecting hand and finger movements, the driver can control most of the internal functions. The driver can point to control and ask what to do. This feature can be used to open or close windows or sunroof. The system uses a gesture camera to capture the hand and finger movements of the inhabitants in three dimensions. [50]

The driver can react by pointing his fingers at what he wants to control. Indicates a window that stands out and can be moved up or down using gestures and voice commands.

For example, if the driver points to a side window, the window will open with the voice command "open".

Figure 28: Pointing to a nearby building to get more information

7.1.1.4. Intelligent Personal Assistant & Wikipedia Interaction

The system can be combined with an intelligent personal assistant. As the next logical development.

Driver entry conversation with the car through an intelligent personal assistant to retrieve information from Wikipedia about what you asked about

For example, the driver asked "Search for Python" or "Search for Barack Obama?" once the assistant searches for an answer online from Wikipedia, it will convert the response into an audible phrase.

7.1.2. External Interaction

Natural Interaction offers the ability to use gestures just by pointing your finger. Drivers can now interact with their immediate surroundings, such as buildings or parking spaces. Pointing a finger at things in his field of vision and issuing a voice command can quickly and easily answer complex queries.

For example, the driver might point and say "What is this building?" "What is the name of this restaurant?" The system will identify the vehicle's position as latitude and longitude using Selenium WebDriver, so with the location coordinates, right elbow angle, and Google API key, the assistant will respond in an audible phrase with information about what the driver asked about.

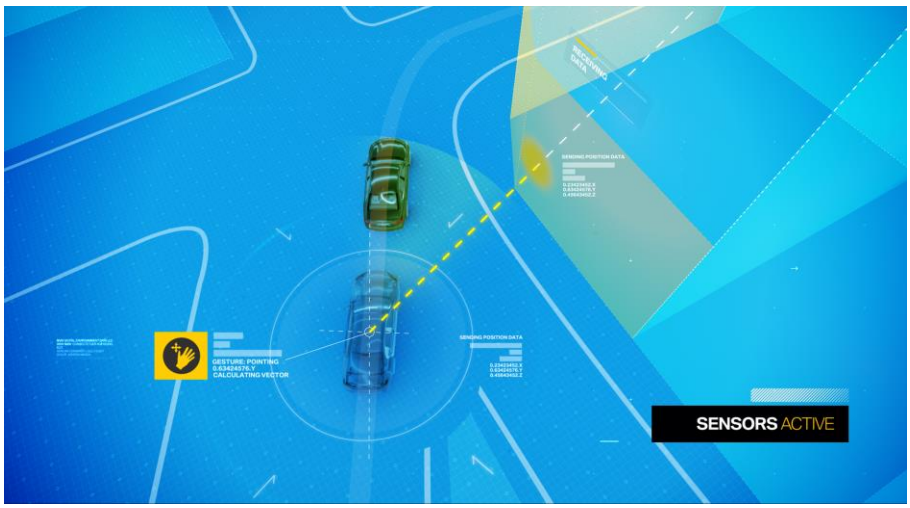

Figure 29: Vehicle Environment Interaction

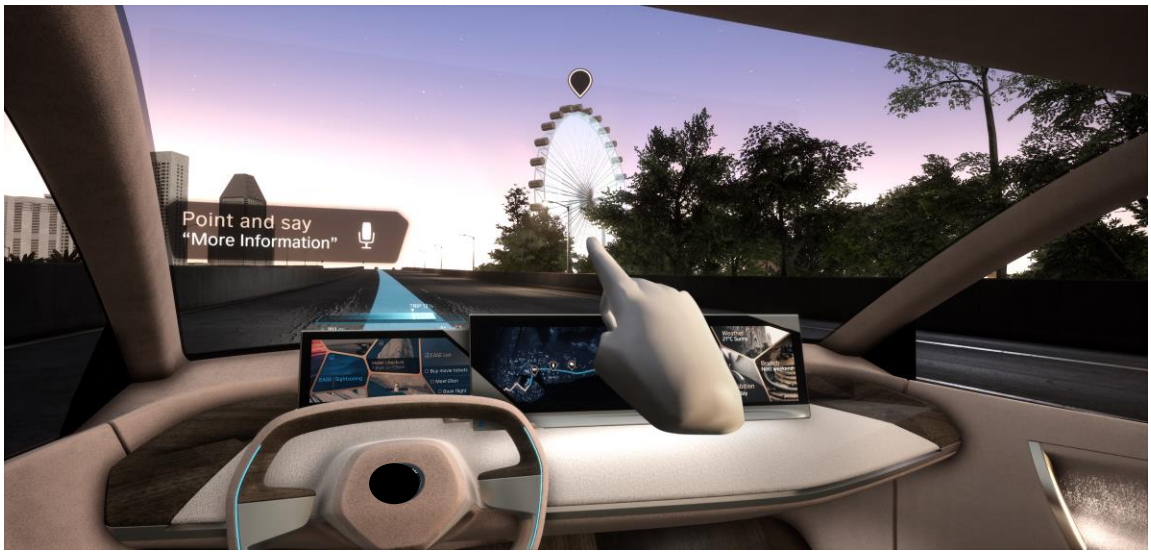

Figure 30: Pointing to a nearby building to get more information

7.2. Summary

Natural Interaction paves the way for the next stage of natural operation in the vehicle and beyond. The free combination of voice instructions, gestures and gaze creates multimodal interaction, based on interpersonal communication.

Natural interaction is also an important step for the future of self-driving cars when interior concepts will no longer be geared solely to the driver's position and passengers will gain more freedom.

Chapter 8.
Results

The performance of all these results of the whole system was calculated using Intel(R) Core (TM) i7-2670QM CPU @ 2.20GHz.

8.1. Face Recognition

Table 4: Face recognition results

	Known person Case	Unknown person Case
Accuracy	99.38%	
Performance	About 4-5 sec	About 13 sec
System login	Allowed	Denied and Notification send to the car owner

8.2. Gaze Tracking

Table 5: Gaze tracking results

Drowsiness Time	**1 sec**	**2.5 sec**
Accuracy	99.3833%	
System Response	Starting the alarm	Auto park the car & Send emergency notification

8.3. Voice Recognition

Table 6: Voice recognition accuracy and performance

Accuracy	95%
Performance	About 1 sec.

8.4. Gesture Recognition

Table 7: Gesture recognition accuracy and performance

	Kinect V2	**OpenPose model**
Accuracy	95%	94.54%
Performance	30 fps	3-4 fps

8.5. Internal Interaction

8.5.1. Question answering

The wake word for this part of the system is “start”, and the sleep one is “end”, where after saying the wake word the system microphone will work similar to before but opening for 5 seconds each instead of only 3 seconds, and when the command search is said the system will search the web “Wikipedia” for the sentence or word after it and return with the acquired results.

8.5.2. Internal Commands

To start or stop this part of the system there 2 commands. First, there is the wake word which is “voice”, and the sleep word which is “mute”, after the wake word the microphone will enter a loop opening frequently for 3 seconds each to catch the command to be executed.

Table 8: Angles referring to each device

Angle	Device
$[20°, 40°]$	Radio
$]40°, 65°]$	AC
$[70°, 90°]$	Right window
$[-90°, -70°]$	Left window

Table 9: Radio commands

Command	Action
“on”	Open the radio
“off”	Close the radio
“forward”	Go to the following channel
“backward”	Go to the preceding channel
“go channel number”	Go to the specified channel number

Table 10: AC commands

Command	Action
“on”	Open the AC
“off”	Close the AC
“increase”	Increase the temperature by 1 degree
“decrease”	Decrease the temperature by 1 degree
“set temperature”	Set the temperature to the desired one

Table 11: Window commands

Command	Action
“open”	Open the window slightly
“close”	Close the window slightly

8.6. External Interaction

This part of the system has a wake word of “wake up”, and asleep one of “sleep”, after the wake word the system microphone works exactly like the last block for opening frequently for 5 seconds, where when the driver points to a location and says “what is this building” the system will search for it and return with its details including its name of this place, ratings, availability, phone number, reviews, and prices.

Chapter 9.
Conclusion and future work

9.1. Conclusion

In conclusion, the current market trend toward artificial intelligence and machine learning. While virtual assistants have been here for quite some time now due to low model accuracy, the hardware used for precision models was complex and the device was power-hungry and expensive. So, the models were cloud-based which were ineffective when it comes to applications in real-time. As hardware becomes cheaper and more energy-efficient. It became possible to put a powerful processor in a moving object while keeping the cost in check, models also became more accurate for acceptable performance. Most car companies are looking to provide an independent car assistant soon to help with the driving experience, save time and money.

Our new system combines the most advanced voice command technology available with expanded gesture control and gaze recognition to enable true multimedia operation for the first time. To quickly and reliably interpret voice instructions as well as gestures, the information that the driver sends to the vehicle is combined in a multimedia manner and evaluated with the help of artificial intelligence. The algorithm responsible for interpreting and improving data inside the vehicle is constantly being improved by using machine learning and evaluating different operating scenarios.

Spoken commands are recorded and processed using Natural Language Understanding (NLU). NLU is a smart learning algorithm, constantly improving. NLU collects and interprets complex speech information, so the vehicle can respond accordingly. Natural Interaction including voice, gestures, and gaze can help facilitate using the full options of the car for a driver more easily and safely with less dispersion while driving leading to a safer drive with a smaller number of accidents, so the driver is now safer and more comfortable.

Thanks to the depth of the vehicle's connections, extensive environmental data, and artificial intelligence natural interaction between the vehicle and the driver were successfully achieved.

9.2. Future work

The development of the interaction between the driver and the vehicle will further advance in parallel. In the future, with the help of artificial intelligence, the system will continue to learn and improved sensor technology will be able to take passengers' emotions into account and integrate them into interaction in a meaningful way. In this way, the interaction between the driver and the vehicle will become more custom and general-purpose. Based on experience and depending on the situation and mood, the smart assistant will be able to decide whether to wait for instructions or offer suggestions to interact proactively.

By connecting digital services, it will be possible to expand the scope of the interaction in the future. For example, when a driver detects a parking spot, he will easily be able to see whether he is allowed to park there or not and how much it costs, then book it and pay it directly without pressing a button.

The smart digital companion is familiar with all vehicle functions and can operate them as needed. Intelligent Personal Assistant also learns routines and habits so that it can be applied in the appropriate context in the future. With each voice command, question, and setting, Intelligent Personal Assistant improves and learns more and more preferences and preferences.

A good way to work on it is to integrate with the autonomous driving system that enables users to get the maximum comfort while enabling the vehicle to fix driver errors. This will create a new platform of smart devices that can connect different types of applications to the surface in the coming years.

References

[1] https://en.wikipedia.org/wiki/Convolutional_neural_network

[2] https://academic.microsoft.com/topic/81363708

[3]https://mc.ai/understanding-cnn-in-python%E2%80%8A-%E2%80%8Ablood-cell-classification/

[4] http://acsjob.com/c8a5f/cnn-input-shape.html

[5] "Concepts and Programming in PyTorch: A way to dive into the technicality",Chitra Vasudevan, BPB Publications, Jun 27, 2018

[6] https://harangdev.github.io/deep-learning/convolutional-neural-networks/22/

[7] "Practical Convolutional Neural Networks: Implement advanced deep learning models using Python", Mohit Sewak, Md. Rezaul Karim, Pradeep Pujari, Packt Publishing Ltd, Feb 27, 2018

[8] "Artificial Intelligence",17th Russian Conference, RCAI 2019, Ulyanovsk, Russia, October 21–25, 2019, Proceedings

[9] https://en.wikipedia.org/wiki/Residual_neural_network

[10] https://tok.fandom.com/wiki/Convolutional_neural_network# "Convolutional neural network | Tree of Knowledge Wiki"

[11] "The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019)", Aboul Ella Hassanien, Ahmad Taher Azar, Tarek Gaber, Roheet Bhatnagar, Mohamed F. Tolba, Springer, Mar 16, 2019

[12] "Convolutional lstm for next frame prediction", http://firass-002-site6.ftempurl.com/nlsoeik3jrhfd/hlnksj35jdssa.php?dftjf326dfhd=convolutional-lstm-for-next-frame-prediction

[13] https://en.wikipedia.org/wiki/TensorFlow

[14] https://opencv.org/about/

[15] https://en.wikipedia.org/wiki/Dlib

[16] https://en.wikipedia.org/wiki/Kinect

[17] https://openkinect.github.io/libfreenect2/

[18] https://github.com/UnaNancyOwen/OpenNI2Sample

[19] https://github.com/UnaNancyOwen/NiTE2Sample

[20] "CMU-Perceptual-Computing-Lab/openpose - GitHub", https://github.com/CMU-Perceptual-Computing-Lab/openpose

[21] "Towards a multi-scenario clinical gait characterization system for neurological diseases JCM Rodrigues" (PDF), repositorio-aberto , 2019

[22] https://cmu-perceptual-computing-lab.github.io/openpose/html/index.html

[23] "igece/SoxSharp: .NET wrapper for SoX. – GitHub", https://github.com/igece/SoxSharp

[24] https://www.ffmpeg.org/about.html

[25] https://www.geeksforgeeks.org/python-text-to-speech-by-using-pyttsx3/#:~:text=pyttsx3%20is%20a%20text%2Dto,a%20reference%20to%20a%20pyttsx3.

[26] "Selenium Webdriver Tutorial with Examples | BrowserStack", https://www.browserstack.com/guide/selenium-webdriver-tutorial

[27] "Nvidia Jetson – Wikipedia", https://en.wikipedia.org/wiki/Nvidia_Jetson

[28]https://www.press.bmwgroup.com/global/article/detail/T0292196EN/natural-and-fully-multimodal-interaction-with-the-vehicle-and-its-surroundings-bmw-group-presents-bmw-natural-interaction-for-the-first-time-at-mobile-world-congress-2019?language=en

[29] https://github.com/ageitgey/face_recognition

[30]https://github.com/kevinam99/capturing-images-from-webcam-using-opencv-python/blob/master/webcam-capture-v1.01.py

[31] "DRIVER FATIGUE AND ROAD ACCIDENTS A LITERATURE REVIEW and POSITION PAPER" (PDF). Royal Society for the Prevention of Accidents. February 2001. Archived from the original (PDF) on 2017-03-01. Retrieved 2017-02-28.

[32] "4.1.03. Driver Drowsiness Detection System for Cars". Retrieved 2015-11-05.

[33][37] Driver's Drowsiness Detecting and Alarming System (PDF). H.M Chandrasena1, D.M.J. Wickramasinghe2 1,2 Faculty of Science, University of Peradeniya, Sri Lanka.

[34] https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/

[35]https://github.com/victordibia/handtracking?fbclid=IwAR0A2r687rnWTwHDOvRCI-iPpgorDT3mOD7hIUvf5mGUrj9eNgjelW39jwU

[36] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In CVPR, 2016.

[38]https://github.com/timctho/convolutional-pose-machines-tensorflow/blob/master/run_demo_hand_with_tracker.py

[39] https://github.com/CMU-Perceptual-Computing-Lab/openpose

[40] https://github.com/ortegatron/hand_standalone

[41] https://github.com/MrEliptik/HandPose

[42] https://github.com/StrongRay/Openpose-Hand-Detection

[43] Petr Altman /"Using MS Kinect Device for Natural User Interface"/ University of west Bohemia/ Department of computer science and engineering/ Master's thesis. (2013).

[44] Bob Corish, Antonio Criminisi, Kenton O'Hara, Abigail Sellen/ Touchless Interaction in Medical Imaging/ Lancaster University/ May 2012.

[45] https://vitruviuskinect.com/angle-calculations/

[46] https://cv-tricks.com/pose-estimation/using-deep-learning-in-opencv/

[47] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam/ "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications"/ 17 Apr 2017.

[48]Plamen Angelov, Robert Kozma/"Proceedings of the 21st EANN(Engineering Applications of Neural Networks) 2020 Conferance" Book/ Jul 26, 2020

[49]https://poolangelofficial.com/2020/02/23/deep-learning-based-human-pose-estimation/

[50] https://www.futurecar.com/3021/BMW-Debuts-a-Natural-Interaction-Assistant-Allowing-Drivers-to-Use-Voice-Gestures-or-Gaze-to-Control-Vehicle-Functions