BLIND READER AND PROTECTIVE SYSTEM

Academic Year: 4th Computer Engineering Department.

Supervisor: Dr. Amr El-Sayed.

List of Student Names and Email:

Name	Email	Group No.
Nancy Mohamed Soliman Mohamed	Nancy.Mohamed.89@h-eng.helwan.edu.eg	A
Asmaa Ehab Ahmed Salah	asmaa99@h-eng.helwan.edu.eg	B
Mario Ashraf Sobhy Hakim	Mario.Sobhy.Hakim.98@h-eng.helwan.edu.eg	D
Mohamed Ali Ragheb El-Sayed	Mohamed.Elsayd.14@h-eng.helwan.edu.eg	D
Mohamed Ali Mohamed Mohamed	Mohamed.Mohamed.818@h-eng.helwan.edu.eg	E

Table of Contents

List of Figures

List of Tables

Abbreviation of Words

1. **AI:** Artificial Intelligence
2. **AMQP:** abbreviation for Advanced Message Queuing Protocol
3. **ANFIS:** Adaptive Neuro-Fuzzy Interference System
4. **ANFIS:** Adaptive Neuro-Fuzzy Interference System
5. **API:** Application Programming Interface
6. **AR:** Augmented Reality
7. **ASCII:** American Standard Code for Information Interchange
8. **ASR:** Automatic Speech Recognition
9. **AVR:** Automatic Voltage Regulation
10. **BiFPN:** Weighted Bi-directional Feature Pyramid Network
11. **BSD:** Berkeley Software Distribution
12. **BVI:** Blind and Visually Impaired
13. **CLI:** Command-Line Interface
14. **CNN:** Convolutional Neural Network
15. **COCO:** Common Objects in Context
16. **CPU**: Central processing unit
17. **CRAFT:** Character-Region Awareness for Text detection
18. **CRM:** Customer Relationship Management
19. **CRNN:** Convolutional Recurrent Neural Network
20. **CSS:** Cascading Style Sheets
21. **CUDA:** Compute Unified Device Architecture
22. **CVPR:** Computer Vision and Pattern Recognition
23. **DLM:** Domain Language Mode
24. **EDI:** Electronic Data Interchange
25. **FOSS:** Free and Open-Source Software
26. **FPN:** Fair Processing Notice
27. **FPS:** Frames Per Second
28. **FRGC:** Face Recognition Grand Challenge
29. **GLVQ:** Generalized Learning Vector Quantization
30. **GPIO:** General Purpose Input/Output
31. **GPS:** Global Positioning System
32. **GPU:** Graphics Processing Unit
33. **GraphQL:** Graph Query Language
34. **GSM:** Globalization Management System
35. **HCI:** Human-Computer Interaction
36. **HOG:** Histogram of Oriented Gradient
37. **HSV:** Hue-Saturation-Value
38. **HTML:** HyperText Markup Language
39. **HTTP:** HyperText Transfer Protocol
40. **IBM:** International Business Machines
41. **IC:** Integrated Circuit
42. **ICR:** Intelligent Character Recognition
43. **IDE:** Integrated Development Environment
44. **INRIA:** National Institute for Research in Digital Science and Technology
45. **iOS:** IDevice Operating System
46. **IoT:** Internet of Things
47. **IOU:** Intersection Over Union
48. **IR:** Infrared
49. **KNN:** K-Nearest Neighbors
50. **LED:** Light Emitting Diode
51. **LSTM:** Long Short-Term Memory
52. **MIPI:** The Mobile Industry Processor Interface
53. **MIT:** Master of Information
54. **ML:** Machine Learning
55. **MMX:** Multimedia extension
56. **MQTT:** Message Queuing Telemetry Transport

57. **NAS:** Neural Architecture Search
58. **NEC:** National Electrical Code
59. **NER:** Named Entity Recognition
60. **NLP:** Natural Language Processing
61. **OASIS:** Organization for the Advancement of Structured Information Standards
62. **OCR:** Optical Character Recognition
63. **OEM:** OCR Engine Mode
64. **OO:** Object Oriented
65. **OpenCV:** Open-Source Computer Vision
66. **ORM:** Object Relational Mapping
67. **OS:** Operating Systems
68. **OSD:** Orientation and Script Detection
69. **PAN:** Personal Area Network
70. **PCA:** Principal Component Analysis
71. **PDF:** Portable Document Format
72. **PIL:** Public Interest Litigation
73. **PIP:** Preferred Installer Program
74. **PSM:** Page Segmentation
75. **PVC:** Permanent Virtual Circuit
76. **RAM:** Random-Access Memory
77. **R-CNN:** Region-based Convolutional Neural Network
78. **REST:** REpresentational State Transfer
79. **RGB:** Blue-Green-Red
80. **RNN:** Recurrent Neural Network
81. **RPC:** Remote Procedure Call
82. **SDHC:** Secure Digital High Capacity
83. **SDK:** Software Development Kit
84. **SFAM:** Simultaneous Frequency and Amplitude Modulation
85. **SFM:** Structure from motion
86. **SIFT:** Scale Invariant Feature Transform
87. **SLU:** Spoken Language Understanding
88. **SMS:** Short Message Service
89. **SOAP:** Simple Object Access Protocol
90. **SOS:** Save Our Souls
91. **SSD:** Single Shot Multi-Box Detector
92. **SSE:** Streaming SIMD Extensions
93. **STL:** Standard Triangle Language
94. **STT:** Speech-to-Text
95. **SURF:** Speeded-Up Robust Features
96. **TCP:** Transmission Control Protocol
97. **TTS:** Text - To – Speech
98. **USB:** Universal Serial Bus
99. **VLC:** Video LAN Client
100. **WHO:** World Health Organization.
101. **Wi-Fi:** Wireless Fidelity
102. **YOLO:** You Only Look Once

Abstract

Visually impaired individuals find it more challenging to move around independently because of their compromised vision. Moreover, a blind person's capacity to navigate in each setting, along with their ability to organize their daily activities are vital to their health and wellbeing. The more saddening fact is that there are tens of millions of visually impaired persons worldwide who must go through such an experience and are dependent on others for their wellbeing and happiness. The encouraging news, however, is that the rapid advancement in technology has seen the innovation of better systems for assisting the disabled, including the blind.

In this project, the main objective is to develop cheap and small hardware while maintaining a well-functioning system for visually impaired individuals. This system can help visually impaired individuals to avoid the obstacles such as people, animals, etc., so that blind people can navigate around safely. In addition, it can measure the distance of the obstacles in front of them, so it warns them to avoid a collision and avoid embarrassment of getting hurt.

Using the most accurate object recognition algorithm, they can detect and recognize the objects around them to give them small glimpse of what is around them in the rooms and streets. Not only is the algorithm accurate but also very fast. There is also a face recognition algorithm that helps blind people to identify the known people around.

Moreover, this system is connected to a mobile interface, using a mobile application that is designed especially for blind people to help them interact with the system. The application helps us to track the visually impaired person using the cellphone's GPS which helps the visually impaired person to make emergency calls in case of emergency.

Chapter 1

Introduction

INTRODUCTION

According to World Health Organization (WHO):

At the global level, in 2014-2019, there are over 2.2 billion people who are visually impaired across the globe (2013), out of which more than 49 million people are blind. India is the second largest population in the world, contributing 30% of the overall blind population. Although there are enough campaigns being conducted to treat these people, it has been difficult to source all the requirements. at least 1 billion – or almost half – of these cases, vision impairment could have been prevented or has yet to be addressed. This 1 billion people include those with moderate or severe distance vision impairment or blindness due to unaddressed refractive error (88.4 million), cataract (94 million), glaucoma (7.7 million), corneal opacities (4.2 million), diabetic retinopathy (3.9 million), and trachoma (2 million), as well as near vision impairment caused by unaddressed presbyopia (826 million), (2013).

In 2015, there were an estimated 253 million people with visual impairment worldwide. Of these, 36 million were blind and a further 217 million had moderate to severe visual impairment (MSVI). The prevalence of people that have distance visual impairment is 3.44%, of whom 0.49% are blind and 2.95% have MSVI, (2013). A further 1.1 billion people are estimated to have functional presbyopia. expected to give rise to a much greater increase in the absolute number of visually impaired people:

1. The global population increased by 38%: from 5.3 billion in 1990 to 7.3 billion in 2015, (2013).
2. The world population aged and the total population over 50 years old almost doubled: from 878 million in 1990 to 1,640 million in 2015, (2013).

In 2021, The World Health Organization has estimated the number of visually impaired globally, based on the latest studies, that there are about 285 million people who suffer from visual impairment (blindness + low vision) worldwide, of whom 246 million people are visually impaired and 39 million people are blind. The Eastern Mediterranean Region accounts for 12.6% of the blindness rate in the world, (2013).

Specifically, in Egypt:

In 2010, The number of people living with blindness in Egypt is more than 800,000 (Basulto, 2014), and the number of people with blindness or low vision in the world, according to estimates by the World Health Organization, is 4% of the world's population; About 37 million people in the Eastern Mediterranean Region suffer from low vision caused by eye diseases or uncorrected refractive errors; Of them, 5.3 million are blind. About 90% (Basulto, 2014) of these live-in low-income countries, The World Health Organization conducted a survey in a number of countries in the region and in Egypt, which showed a rapid assessment of preventable blindness in five governorates, namely Beni Suef, Fayoum, Kafr El-Sheikh, Minya, and Sohag, and the prevalence

of blindness cases among the population of the age group over fifty in large proportions. The results of this survey showed that Beni Suef governorate occupies the first place among the five governorates in terms of the prevalence of blindness, which reached more than 10% among the population over the age of fifty. It is followed by Minya governorate with a percentage that also exceeds 10% of the population in the same segment, while the incidence of blindness among residents over fifty in Sohag, Fayoum and Kafr El-Sheikh governorates ranged between 6% and 7.5% (Basulto, 2014).

In 2015, Egypt, which has approximately 1 million people blind and 3 million visually impaired. Nearly 60% of the visually impaired in Egypt (Basulto, 2014).

In 2021, In Egypt, the number of blind people in Egypt is estimated at about 3.5 million blind Egyptians, and the number of blind people enrolled in the various stages of education is about 37,000 (Basulto, 2014).

Vision disability is one of the top 10 disabilities among adults 18 years and older and one of the most prevalent disabling conditions among children. Vision loss causes a substantial social and economic toll for millions of people including significant suffering, disability, loss of productivity, and diminished quality of life. the blind people always face difficulties in their daily life. they always need help from another person to move from one place to another or to read anything, sometimes this make them feel embarrassing, sadness and that they are burden to the others, or they need a stick or a dog to guide them when moving to avoid hitting anything, but this method is traditional and unsuccessful most of the time. And they also need another person to be able to read what is in front of them, whether it is a sign or the prices of goods in the supermarket or money. the purpose of that survey and project is make the life of that people easier, simpler, and happier. And help them to do their needs by own.

It is the era of artificial intelligence, and it has gained immense traction due to a large amount of data and ease of computation (Basulto, 2014). Using artificial intelligence, it is possible to make these people's life much easier and happier. The goal is to provide a "secondary sight" until they have enough resources required to treat them. People with untreatable blindness can use this to make their everyday tasks much easier and simpler. The scope of this research is to reach a solution and to help the blind or visually impaired people move without any help from any body and ability of read easily and they can make use of this device to enhance their knowledge and read anything in front of them.

PROBLEM STATEMENT

Visually impaired people face many difficulties and reading is one of the difficulties they face and when they engage in self-navigation in an environment alien to them. In fact, physical movement is one of their biggest challenges. Besides, while they are wandering or walking in a crowded lane, it can be quite difficult because they can't recognize the obstacles around them.

One of the current problems for visually impaired people with lane travel is that they cannot detect whether they need to turn left or turn right when reaching the end of the lane using only a walking stick. According to (M.F.Saaid, A.M.Mohammad, & Ali, 2016). The reason why visually impaired people do this is that they cannot predict the obstacle that is far away from them while they can only use a walking stick to discover the area around them.

Daily Life Problems, Struggle and Challenges Faced by Blind People

1. Navigating Around Places

The biggest challenge for a blind person, especially the one with the complete loss of vision, is to navigate around places. Obviously, blind people roam easily around their house without any help because they know the position of everything in the house. People living with and visiting blind people must make sure not to move things around without informing or asking the blind person. Commercial places can be made easily accessible for the blinds with tactile tiles. But unfortunately, this is not done in most of the places. This creates a big problem for blind people who might want to visit the place.

2. Finding Reading Material

Blind people have a tough time finding good reading materials in accessible formats. Millions of people in India are blind but we do not have even the proper textbooks in braille, leave alone the novels and other leisure reading materials. Internet, the treasure trove of information and reading materials, too is mostly inaccessible for the blind people. Even though a blind person can use screen reading software, but it does not make the Internet surfing experience very smooth if the websites are not designed accordingly. Blind person depends on the image description for understanding whatever is represented through pictures. But most of the time, websites do not provide clear image description.

3. Arranging Clothes

As most of the blind people depend on the objects' shape and texture to identify them arranging the laundry becomes a challenging task. Although a majority of blind people device their own technique to recognize and arrange at least their own clothes but it still is a challenging chore. This becomes a daredevil task if it's about pairing and arranging the socks. All this is because recognizing colors is almost impossible for the persons with total blindness.

4. Overly Helpful Individuals

It is good to be kind and help others. But overly helpful individuals often create problems for the blind person. There are lots of individuals who get so excited to help a disabled person that they forget even to ask the person whether she needs help or not. A blind person might be doing something painfully slow (from your perspective), but you should not hurry in doing the work without asking the person properly. You might end up creating some trouble for the blind person.

5. Getting Devices to Become Independent

The most valuable thing for a disabled person is gaining independence. A blind person can lead an independent life with some specifically designed adaptive things for them. There are lots of adaptive equipment that can enable a blind person to live their life independently, but they are not easily available in the local shops or markets. Refreshable Braille Display is an example of such useful devices. A blind person needs to hunt and put much effort to get each equipment that can take them one step closer towards independence.

Everyone faces challenges in their life… blind people face a lot more. But this certainly does not mean that you can show sympathy to blind persons. They too, just like any individual, take up life's challenges and live a normal life, even if it does not seem normal to the sighted individuals.

6. Access to information

The major sensory organ of a person is their eyes. One glimpse around us is enough to make us realize how visual is most of the information in our environment. Timetables in train stations, signs indicating the right way or potential danger, a billboard advertising a new product in the market, these are all the visual types of information we all come across in our daily life. Most of this information is inaccessible for the blind and the visually impaired, inhibiting their independence since access to information signifies autonomy.

7. Overly helpful individuals

It's very common for sighted individuals, strangers, friends or family, to be overly excited to help a visually impaired person. Very frequently, this behavior holds the assumption that the blind or low vision individual requires assistance, although this might not reflect reality. Blind people might perform a regular task slower but that doesn't mean they're incapable of completing it. Rushing to help the visually impaired without asking or being asked to do so, might make them feel helpless instead of independent. Moreover, not allowing a visually impaired individual perform a task by themselves, does not give them the room to learn how to do so independently.

8. Societal stigma

Being blind in a world suited for the sighted, it means there will be multiple normal mishaps. Stumbling upon an office chair that wasn't neatly tucked under the desk or knocking a glass off the table because it was left right on the edge, are small accidents that can happen and that's okay. However, such mishaps tend to be perceived by sighted individuals as the inability of the visually impaired to perform tasks, while, in reality, they stem from the inaccessibility of our world. Blindness or low vision does not indicate the intelligence of the individual nor how sad their life is. Just because the sighted cannot imagine their world without vision does not mean that the visually impaired have a sad or unhappy life because of their visual condition.

9. Finding and keeping a job

Work is a whole different matter if you're visually impaired. Considering the lack of accessible work and working spaces, one can already imagine why hiring a visually impaired individual would be considered a liability for a company. This has a negative impact on the confidence and emotional well-being of the visually impaired, while it totally cripples their economic independence. Having little to no opportunity to support oneself, blind or low vision individuals are incapacitated from their independence.

10.Leisure

The lack of accessibility for the visually impaired is central to a number of the issues the blind or low visual individuals face. Leisure is another one on the list. There is a limited number of

inclusive/accessible activities for the visually impaired, which are as simple as a museum visit. Moreover, accessible books are not abundant either. According to the World Blind Union, "more than 90% of all published material is not accessible to the blind or partially sighted." The internet, as in the new era we all surf the internet for fun, is not fully accessible either since numerous websites disregard their visually impaired visitors and do not curate content that is accessible for the blind and low vision individuals. Considering these points and many more that are not listed here, one can clearly see how limited leisure options there are for the visually impaired.

Conclusion:
Often living in isolation Considering all the above, it's not a surprise that living with a visual impairment might signify, often, living in isolation. Dealing with sight loss, already, is a challenge in itself. The lack of emotional support at diagnosis centers, the limited accessibility to activities and information, the societal stigma, and the lack of unemployment, are all factors frequently leading blind or low vision individuals in isolation. This last point illustrates how the problem for the visually impaired is not their blindness or lower vision in itself but their segregation from anyone else.

SOLUTIONS

Objective

In this project, the main objective is to develop a cheaper price but will still maintain a good functional system for visually impaired individuals. This system can help visually impaired individuals to avoid the obstacles such as people and animals on the corridor same with them, and it also can provide the distance of the obstacles in front of them. The aim of this project is to improve the visually impaired individuals' ability in finding the direction in the corridor while they are walking rather than just rely on the walking stick to detect all the obstacles manually and waste their time in finding the exact direction that they want to head to.

General Objectives

- To simplify the physical movement of a visually impaired person.
- To replace the conventional walking cane with smart walking system what detect obstacle.
- To make the blind person safe while they are walking.
- To simplify reading process of a blind person.

Specific Objectives

- To design a walking system that detects obstacles using the camera.
- To design a reading system for a blind person using the camera.
- To make vibration alert to the person.
- To positioning and sending emergency messages.

PREVIOUS PROJECTS

I. Third Eye for Blind

The proposed guidance system can determine the obstacle distance, in addition to the material and shape characteristics of the obstacle. Furthermore, the system can name some of the detected objects and also it can read out loud eBooks for blind people who cannot read books. Also, the system can Face Recognition and differentiate between known people and unknown ones; Object Recognition here not only inform the blind that there is an object but tells what is that object using AI Algorithms; there is a text Reader that helps the blind reading Product Labels and book, etc., (Mishra, Sharma, & Ramaiah, 2020).

System description:

This smart stick is an electronic walking guide that has four ultrasonic sensors. These four sensors are used for obstacle detection and are placed on the front and side of the stick. The smart stick gives the output through an earphone.

Raspberry Pi:

- A low-cost high-performance computer which can be plugged in TV and monitor - and can be used as a computer even.
- Its CPU is 700Mhz single core ARM1176JZF-S.
- It has 4 USB ports.
- It has a dual-core video core iv multimedia co-processor.
- The size of its RAM is 512 MB.
- It has a micro SDHC slot for storage.
- The power rating of raspberry pi is 600mA i.e., 3.0W.
- It has 17 GPIO plus the same specific functions.
- The raspberry pi works as the computer of the smart walking stick.

Ultrasonic Sensor

The ultrasonic sensor is a type of sensor that detects an object using sound waves. Its principle is like that of radar or sonar, which generates high-frequency sound waves and receives them back. Sensors calculate the distance using the time taken for the reception of the echo signal sending the signals and receiving back the echo signals to determine the distance of an object.

Camera Module

This module takes pictures and sends them to the microcontroller. The frequency of taking images can be altered depending on the program and usage. It can be used also to take high-definition videos. It even supports low light clicking and is capable of 1080p30, 720p60 and VGA90 video modes.

Software requirements

1. **OpenCV:** OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. Let's start the chapter by defining the term "Computer Vision".
2. **Computer Vision:** Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modelling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields:

- Image Processing − It focuses on image manipulation.
- Pattern Recognition − It explains various techniques to classify patterns.
- Photogrammetry − It is concerned with obtaining accurate measurements from images.

Algorithm

1. **Optical Scanning**:

The optical Scanning process involves capturing a digital image of the original document. The OCR optical scanners that are used will convert the light intensity into grey levels. This process is called Thresholding. Thresholding converts a multilevel image into a bi-level image of black and white.

2. **Location and Segmentation**:

Segmentation involves the isolation of characters or words. Location of the text is done via pixels with x and y coordinates.

3. **Pre-processing**:

The resulting image from the process of scanning may contain some amount of noise. Depending on the resolution of the scanner the characters may be broken or smeared. Hence pre-processing can be done to smooth the digitized character. In addition to smoothing pre-processing also involves the normalization of the characters.

4. **Feature Extraction**:

This technique is used for capturing the essential characteristics of the symbols. Feature extraction is done by matching the matrix containing the input character with a set of prototype characters that represent each possible class.

5. **Recognition**:

The recognition is the process of identifying each character and assigning it to the correct character class.

Block Diagram

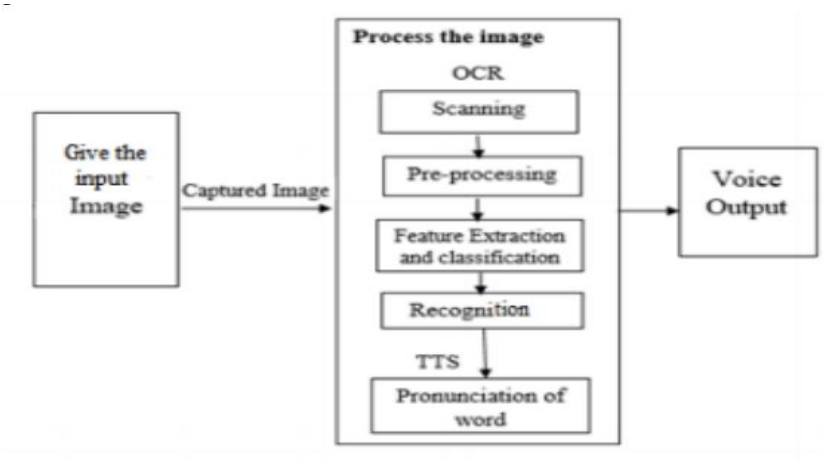


FIGURE 1-1 BLOCK DIAGRAM OF THIRD EYE FOR BLIND

Flowchart

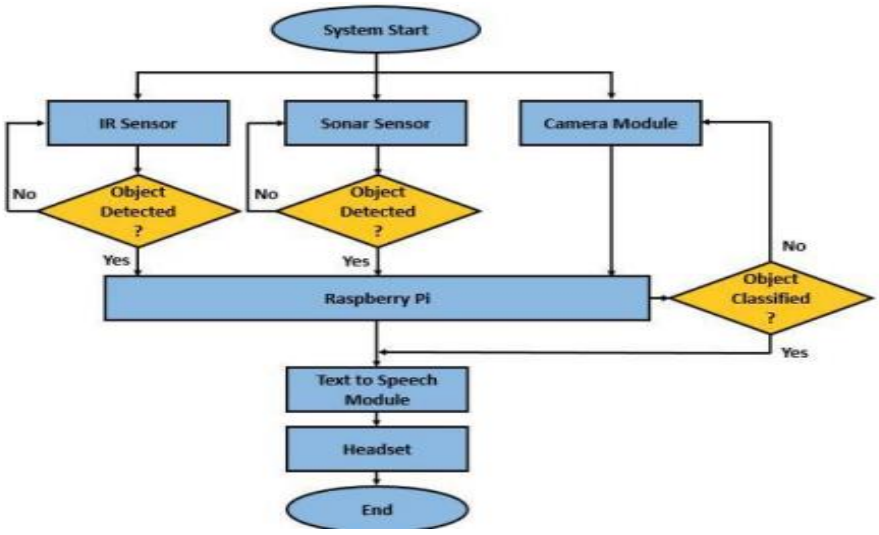


FIGURE 1-2 FLOW CHART OF THIRD EYE FOR BLIND

II. Third Eye for Blind Person

The third eye for Blind task is a development that helps the outwardly debilitated individuals to move around and move between Different places with speed and certainty by knowing the adjacent hindrances utilizing the assistance of the wearable band which delivers the ultrasonic waves which inform them with the inbuilt voice assistant.

Object Detection API is the framework for creating a deep learning network that solves object detection problems to identify the object in front of them. Distance Measurement-Using ultrasonic

we can measure the distance between the object and the user. The object can be anything like a vehicle, chair, person, table etc. Speech Recognition: The blind cannot use the keyboard on an Android Smartphone or even if he/she can type then it is obvious that it will take more time than the normal person. So, to take the input from the user we will use the speech recognition module which is a python module that converts the speech into text and then based on the text it will take actions and control other modules based on the user's command (Singh, Kamat, & Chisti, 2021).

System description

- In this system, we are developing a navigation system for blind persons. This is very easy to use and work as a navigator for blind people to easily navigate.
- In this system, the ultrasonic or sensor will detect the object and gives sound (object 'beep sound') and camera scan the object using object detection technique and predict the object and by using speech recognize the object name is converted into sound and the client can know the object by the help of headset.
- The object and person name and data are stored in the module and if the data is not present it will simply be said no data image present gives a beep sound.
- In this system we are using some hardware and software components which are following Also we add some extra features like distance measurement technique to identify how far the distance between the object and the client and voice assistant for various extra features.

Software's and Technique

- Pi operating system.
- Object detection Technique
- Distance Measurement Technique
- Speech Recognition.

Hardware Components.

- Raspberry pi 4 module.
- Ultrasonic sensor.
- Headset.
- Pi cam (camera).
- 5 mm LED: Red.
- Slide Switch.
- Female Header.
- Male Header.
- Jumper wire.
- Power bank.
- 1kΩ Resistor.
- 2kΩ Resistor.
- Some elastics and stickers.

Raspberry Pi

- A low-cost high-performance computer which can be plugged in TV and monitor and can be used as a computer even.
- Its CPU is 700Mhz single core ARM1176JZF-S.
- It has 4 USB ports.
- It has a dual-core video core iv multimedia co-processor.
- The size of its RAM is 512 MB.
- It has a micro SDHC slot for storage.
- The power rating of raspberry pi is 600mA i.e., 3.0W.
- It has 17 GPIO plus the same specific functions.
- The raspberry pi works as the computer of the smart walking stick.

Pi camera

The Pi camera module is a portable lightweight camera that supports Raspberry Pi. It communicates with Pi using the MIPI camera serial interface protocol.

Ultrasonic Sensor

The ultrasonic sensor consists of a transmitter, receiver, and transceiver. The transmitter converts an electrical signal into sound waves. The receiver converts the sound waves into an electrical signal again.

Software requirements

Object Detection

The tensor Flow object detection API is the framework for creating a deep learning network that solves object detection problems.

Distance Measurement

Using ultrasonic we can measure the distance between the object and the user. The object can be anything like a vehicle, chair, person, table etc. This will be a relative measure given that the picture can be of different angles and perspectives.

Speech recognition

The blind cannot use the keyboard on an Android Smartphone or even if he/she can type then it is obvious that it will take more time than the normal person. So, to take the input from the user we will use the speech recognition module which is a python module that converts the speech in text and then based on the text it will take actions and control other modules based on the user's command.

Algorithm

Object detection using tensor flow

1. Algorithm based on classification they are implemented in two stages:
    a. First, they select regions of interest in an image.
    b. Second, they classify these regions using a convolution neural network.
2. Algorithm based on regression instead of selecting interesting parts of an image, they predict classes and bounding boxes for the whole image in one run of the algorithm. Two best-known examples from this group are the YOLO (You Only Look Once) and SSD (Single Shot Multi- Box Detector).

Distance measurement

Using ultrasonic we can measure the distance between the object and the user. The object can be anything like a vehicle, chair, person, table etc. This will be a relative measure given that the picture can be of different angles and perspectives. To draw the lines around the objects we need to import some inbuilt modules from PIL import Image, Image Draw Import tools.

Block Diagrams

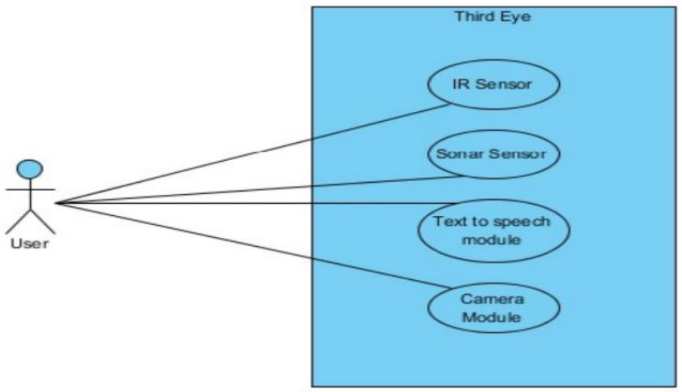


FIGURE 1-3 BLOCK DIAGRAM OF THIRD EYE FOR BLIND PERSON

Flowchart

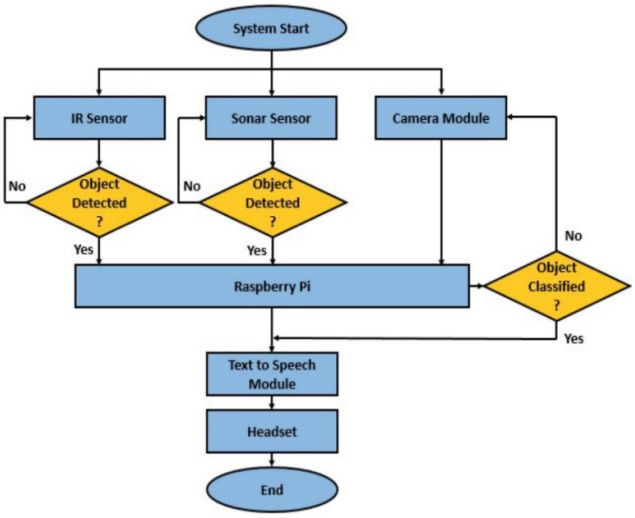


FIGURE 1-4 FLOW CHART OF THIRD EYE FOR BLIND PERSON

III. Smart Stick for Visually Impaired

The objective of the project is to assist the blind person to navigate conveniently and independently (Mind, Palkar, Mahamuni, & Sahare, 2021).

The Hardware unit consists of Raspberry pi, mounted on a PVC pipe to be used as a stick, which is a processing device with ore features like Wi-Fi, more memory storage, etc. The Wi- Fi module in Raspberry pi used to connect with android applications on the phone. Users will Enter their destination location into the app using Voice Command Assistance on Android Phones.

The obstacle detecting sensors, speakers are attached to the Raspberry Pi. The programmed Raspberry Pi then gives the respective instruction about obstacles sensed by sensors. The voice output is generated through TTS (Text to Speech) IC.

The software part consists of an android application installed in the user's phone, which is synced with the hardware module. The user will set the location through the app with voice command, then the GPS will navigate him to the destination set, with the information of obstacles faced by him on the way. The system also has an emergency switch.

When a user faces any difficulty or is in a dangerous situation the emergency message will be sent to the guardian. Voice interaction will make it easier for a blind person to navigate.

System description

This smart stick is an electronic walking guide that has four ultrasonic sensors. These four sensors are used for obstacle detection and are placed on the front and side of the stick. The smart stick gives the output through an earphone.

Raspberry Pi

- The low-cost high-performance computer can be plugged in TV and monitor and can be used as a computer even.
- Its CPU is 700Mhz single core ARM1176JZF-S.
- It has 4 USB ports.
- It has a dual-core video core iv multimedia co-processor.
- The size of its RAM is 512 MB.
- It has a micro SDHC slot for storage.
- The power rating of raspberry pi is 600mA i.e., 3.0W.
- It has 17 GPIO plus the same specific functions.
- The raspberry pi works as the computer of the smart walking stick.

Vibration Motor

A vibration motor is used to inform the user about an obstacle detected by the ultrasonic sensors.

Ultrasonic Sensor

An ultrasonic sensor is a type of sensor that detects an object using sound waves. Its principle is like that of radar or sonar, which generates high-frequency sound waves and receives them back. Sensors calculate the distance using the time taken for the reception of the echo signal sending the signals and receiving back the echo signals to determine the distance of an object.

GSM/GPS 800L

When the GSM modem receives a message, the microcontroller will process the message with the keyword saved in it. Then, it will get the location of the stick from the GPS modem and transmit the location to the GSM modem to respond to the sender. In case of an emergency, the user of the stick can press the emergency button the microcontroller accesses the location from the GPS modem and transmit the location to the GSM modem which will send an SMS message to all saved numbers in the microcontroller.

Buzzer

A transducer (converts electrical energy into mechanical energy) that typically operates A buzzer is in the lower portion of the audible frequency range of 20 Hz to 20 kHz. This is accomplished by converting an electric, oscillating signal in the audible range, into mechanical energy, in the form of audible waves. The buzzer is used in this research to warn the blind person against an obstacle by generating sound proportional to the distance from the obstacle.

Software requirements

Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, it is designed specifically for Android development. It is available for download on Windows, Mac OS and Linux based operating systems. Android is an open-source and Linux-based operating system for mobile devices such as smartphones and tablet computers.

Algorithm

STEP 1: Start.
STEP 2: connect Raspberry pi to android phone through Wi-Fi.
STEP 3: set location on the android application of the phone.
STEP 4: GPS navigation will direct through speaker and vibration alert.
STEP 5: if the sensor detects obstacles, it will vibrate stick and give output through speaker if obstacle at right it will say right through the speaker, like left and front.
STEP 6: If a person gets emergency, he will press the SOS button on the stick it will send an emergency message to the person family with location.
STEP 7: if a person not finding his stick, he will find stick using the application.
STEP 8: stop.

Block Diagram

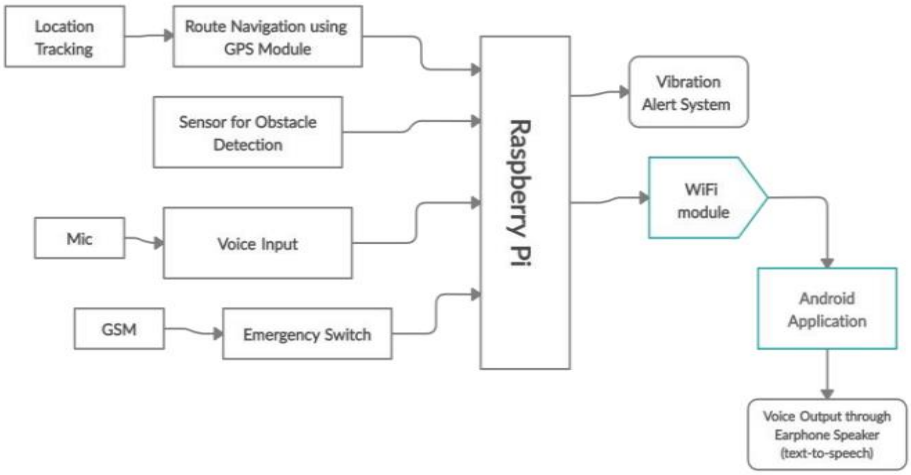


FIGURE 1-5 BLOCK DIAGRAM SMART STICK FOR VISUALLY IMPAIRED

Flowchart

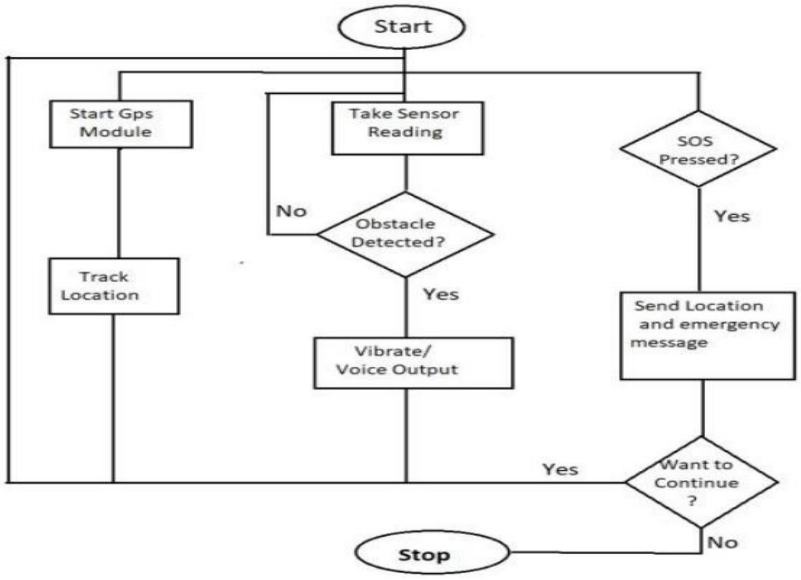


FIGURE 1-6 FLOWCHART OF SMART STICK FOR VISUALLY IMPAIRED

IV. Smart Reader for Blind and Visually Impaired (BVI)

This system aims to develop cost-effective assistive technologies for blind people and provide a greater degree of independence in their daily activities. The main advantage of this research is to detect the text and give the voice output for the visually impaired. It will help the blind to read and acquire knowledge from all the printed material without converting the same into Braille format (M.N., H.V., S., J., & H.C., 2019).

Smart Reading assistance for the blind and visually impaired is carried out. The existing systems have many drawbacks. Here we propose a new idea where the system provides an autonomous page-turning mechanism and interactive dictionary querying feature, ultimately giving a feeling of comfort for BVI.

Hardware Components

- USB Camera – as an input device for capturing the image of the book's page.
- Raspberry Pi 3 Model B – as a processing unit.
- Speaker/Headphone – as an output device for listening in to the speech output. 4) Servo Motors – as actuators for the roller wheel, lifter, and turner arms.
- Microphone – as an input device for the dictionary query feature.
- Push Buttons – as an interrupt button for BVI user interaction.
- Bench support –rectangular plywood for hosting the mechanism, Camera, and RPI system all together tightly.
- Book – the main source of knowledge, for capturing the image and turning mechanism.
- Monitor – as a display device for Verification and Debugging.

Software requirements

The raspberry pi is instructed through the Python 2.7 programming platform. The various modules installed on python are:

- Open CV library –for image capture and preprocessing
- Google Cloud Vision API or Python- Tesseract – Image to Text conversion engine.
- GTTS – google text to speech conversion engine.
- Python-VLC – sound player.
- Py Dictionary – for dictionary query feature.
- Speech Recognition – for speech to text conversion.

Block Diagram

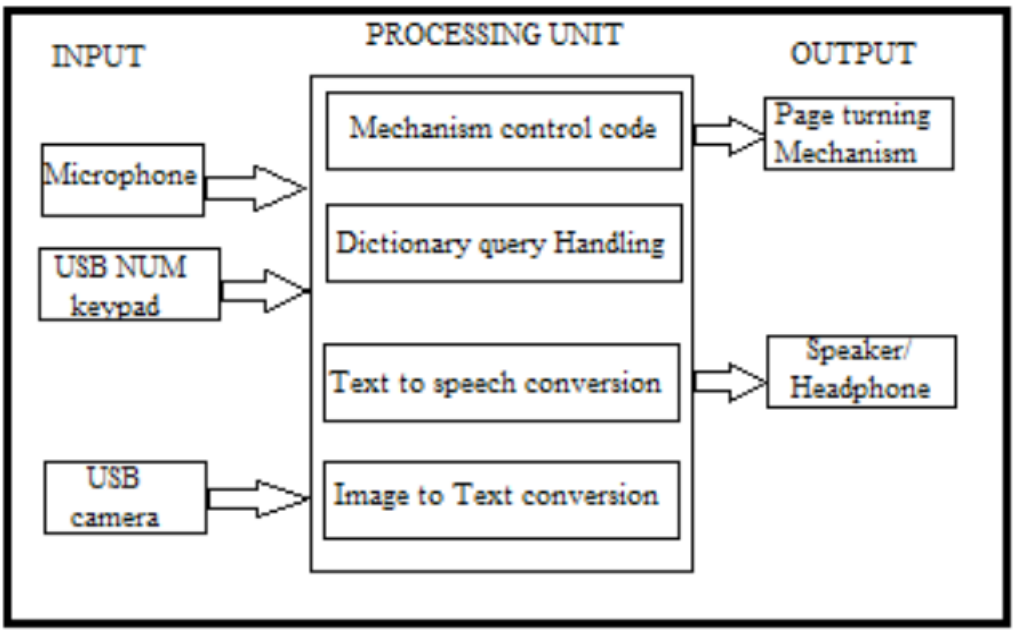


FIGURE 1-7 BLOCK DIAGRAM OF BVA

V. Eye Ring Project

Eye Ring is designed to support operation by both sighted and visually impaired users in the same fashion. It is worn on a finger and still allows one to use the hand and finger for feeling and holding The Eye Ring is an input device consisting of a camera mounted on a ring (typically worn on the index finger) with an embedded processor and wireless connection to a computation device (typically a smartphone). We employ several computer vision techniques to recognize objects or locations based on one or more images taken by the ring. The Eye Ring system also has a speech recognition service to enable voice commands and speech output (as well as screen output) as the means of communicating information. This platform enables a whole series of applications in which a user may acquire information or control objects/spaces in their proximity. For example, a blind user could simply say ‘currency’ and point at a currency note to hear its value it (Singh, Kamat, & Chisti, 2021).

Hardware Components

- Jpeg camera.
- AVR processor.
- Bluetooth module.
- Polymer lithium-ion battery.
- Push button switch.

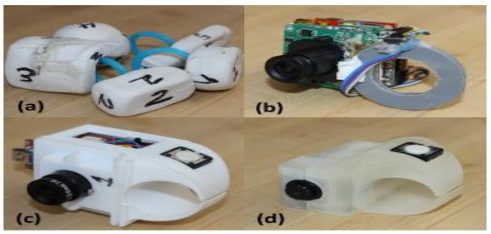


FIGURE 1-8 EYE RING PROJECT

Software's and Technique

A Bluetooth communication module that connects Eye Ring with a smartphone running Android 2.2 or with a Notebook computer running Windows 7. These modules receive binary image data from the ring, as well as button click events. Some of the computer vision algorithms (e.g., currency recognition, tag recognition).

Problems

Need to optimize the design, especially in terms of faster image acquisition and smaller packaging.

Future work

As a real-time video feed from the camera, increased computation to perform low-level computer vision tasks such as edge detection or optical flow, and additional sensors like gyroscopes.

Block Diagram

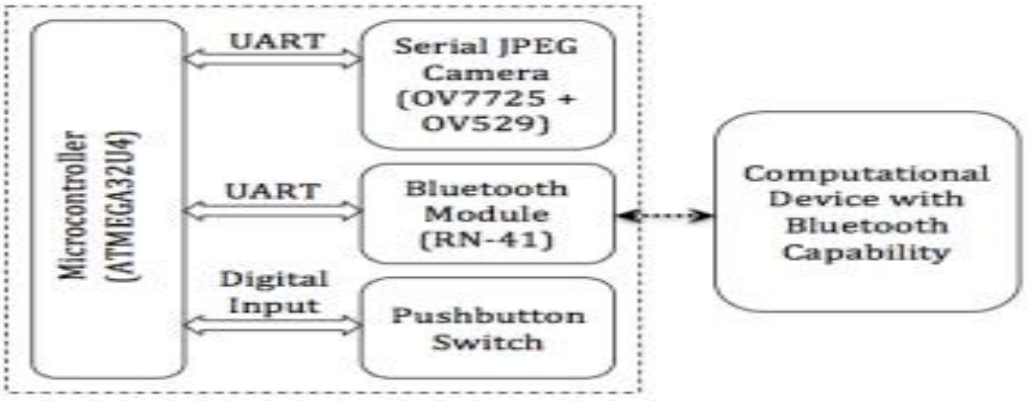


FIGURE 1-9 BLOCK DIAGRAM OF EYE RING

VI. A Text Recognizing Device for Visually Impaired

Visually impaired people can easily read the text without anybody's help They can also use this to read the text written on the tablet, books, board etc. just by wearing this device visually impaired people can perform text reading operations as common peoples read the books (Nanayakkara, Shilkrot, Yeo, & Maes, 2013).

Software Algorithm

Text extraction algorithm. We have to provide the input to the text extraction algorithm a closed up printed text image is given as the input to that algorithm.

Advantage

Detect the text and give the voice output for the visually impaired. if it misses any line the device vibrates and gives the voice output.

Disadvantage

Just reads the text and does not specify whether it is a headline are sub-headline. The camera does not auto-focus it takes time to focus on the text.

VII. Smart Blind Stick

Blind people always need help while travelling around to their destination. In this paper, we have developed a smart stick which helps the blind person to move around confidently without any hesitation. This smart stick is cheap and fast. The stick consists of 2 ultrasonic sensors for obstacle detection and pothole detection. The android application is linked with the smart stick through Bluetooth adapter (Zhou, 2019).

The application also consists of real time navigation system with speech recognized destination, calling and text message sending with speech recognition etc.

Hardware Components

- Ultrasonic sensors.
- Android phone.
- Button.
- IR sensor.
- Arduino.

Software's and Technique

- Obstacle Detection.
- Pothole Detection.
- Android Application.

Software Algorithm

Obstacle Detection

The algorithm used for obstacle detection is as follows:

STEP 1: Dist_front = distance received from sensor 2 or front sensor
STEP 2: If (dist_front>=100 && dist_front<=200) alert that obstacle is ahead. Go to step 5
STEP 3: Calculate dist_right and dist_left i.e., distance from right and left sensor respectively.

STEP 4: If dist_right<100 && dist_left>100 alert Turn left else if dist_left<100 && dist_right>100 alert Turn right else if dist_left<100 && dist_right<100 alert Path is blocked else alert Turn anywhere go to step 1
STEP 5: check dist_front again for closeness
STEP 6: if (dist_front<100) alert obstacle is very close in front go to step 3

Pothole Detection

1. The pothole sensor attached at the bottom of the stick, facing towards ground, sends reading of the time required for the ultrasonic waves to reflect back from the ground, which is then converted to distance using the distance formula.
2. Speed of sound in air =340 m/s
3. Distance= (speed*time)/2
4. Distance is divided by 2 is done because initial distance received is for sending signal plus receiving signal.
5. Initially values are used to calculate the threshold value and later each distance value calculated is compared with the threshold to check for pothole. Pothole
6. detection detects a pothole on users' path by comparing each new value with the calculated threshold value. Pothole detection calculates a threshold value by measuring the stick users' patterns of using the stick (since each person's height is different and sticks holding point is also different), which is then used for detecting a pothole on the user's path. Initial ten values are used for calibration of the threshold value for pothole detection. The value within certain limit is considered for calibration as users' misjudgments might lead to ambiguous results.
7. The valid results are then summed to calculate their average value and the largest value in the set is also recorded. The average value is an indicator of the height to which the user generally lifts the stick while commuting, whereas the highest value is the maximum deviation from the average during the calibration. The difference between average and the maximum value gives a maximum fluctuation value, which then is doubled and added to the average to calculate threshold i.e., error is also considered. This approach increases the accuracy of the threshold and provides the Arduino with a value indicating the maximum possible distance from the ground.
8. avg_value= (sum of 10 values)/10
9. max_value= maximum of initial 10 values
10. fluct_value= max_value-avg_value.
11. Threshold= 2*fluct_value+avg
12. The Arduino calculates distance from ground with each loop, using the readings from the ultrasonic sensor, and compares the new value with threshold value calculated previously to check for potholes. A value greater than the threshold value indicates the possibility of presence of a pothole ahead of the user.

Android Application

The android application is connected with the stick through the Bluetooth adapter. There is one button placed on the stick. If the user feels that he/she is in emergency, then they can press the button and then using speech they can tell the android application to either call or send text message to the desired person. They can even send location to their relatives using this feature. One more feature over here is that the user can use gesture feature of double tapping on screen to start the real time navigation feature. They will use speech to text feature to take the destination value from user so that they can give the shortest path to the end user.

V. RESULT

Sr. No.	Obstacle from	Output	Range(cm)	Result
1	Front	Buzzer	145	Accurate
2	front	Buzzer	78	Accurate
3	front and left	Buzzer	60	Accurate
4	front and right	Buzzer	82	Accurate
5	front, left and right	Buzzer	59	Error
6	front, left and right	Buzze	72	Accurate

TABLE 1-1 OBSTACLE DETECTION

Detection Here there are some test results for the different obstacles in front of the stick. Using these sensors, we have seen the results being recorded at different situations.

Sr. No.	Destination	Voice	Text	Navigation
1	Swargate	Swargate	Swargate	Started
2	FC Road	FC Road	FC Road	Started
3	Kothrud	Kothrud	Katakir	Not started

TABLE 1-2 NAVIGATION

Here are some test cases for navigation feature being used in this application.

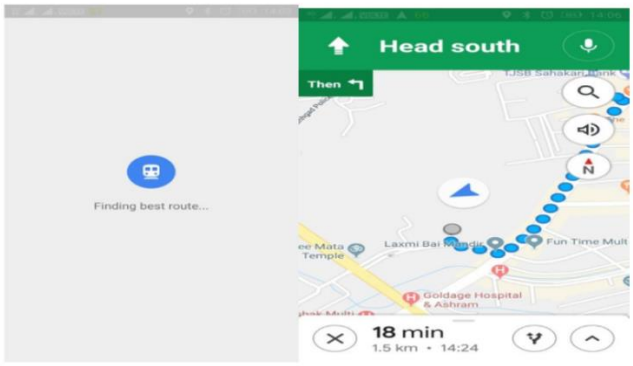


FIGURE 1-10 SHOWING THE NAVIGATION FEATURE BEING USED BY THE BLIND PERSON

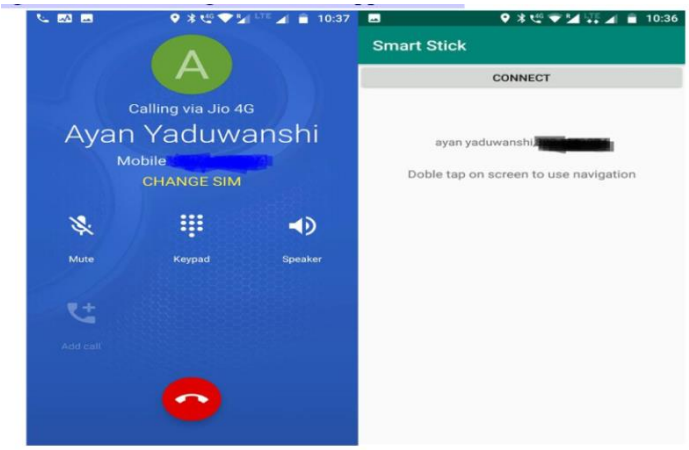


FIGURE 1-11 SHOWING THE CALLING FEATURE BEING KEPT IN THE APPLICATION

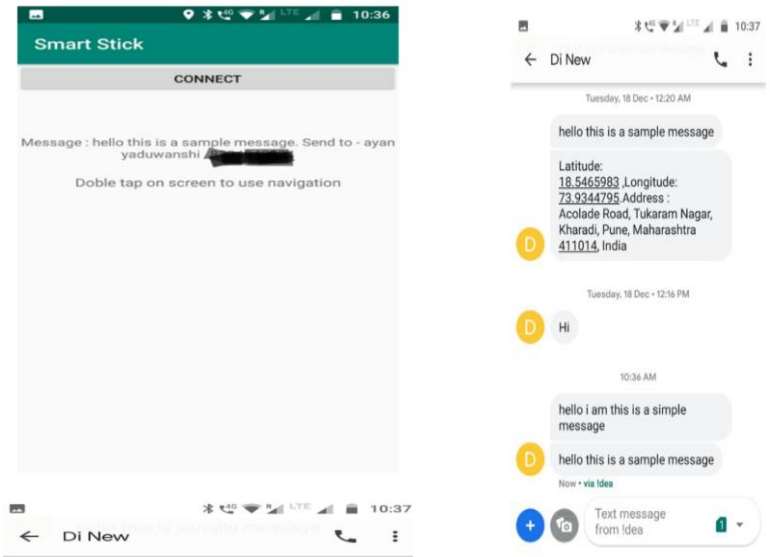


FIGURE 1-12 SHOWING THE TEXT MESSAGE BEING USED WITH THE HELP OF ANDROID PHONE

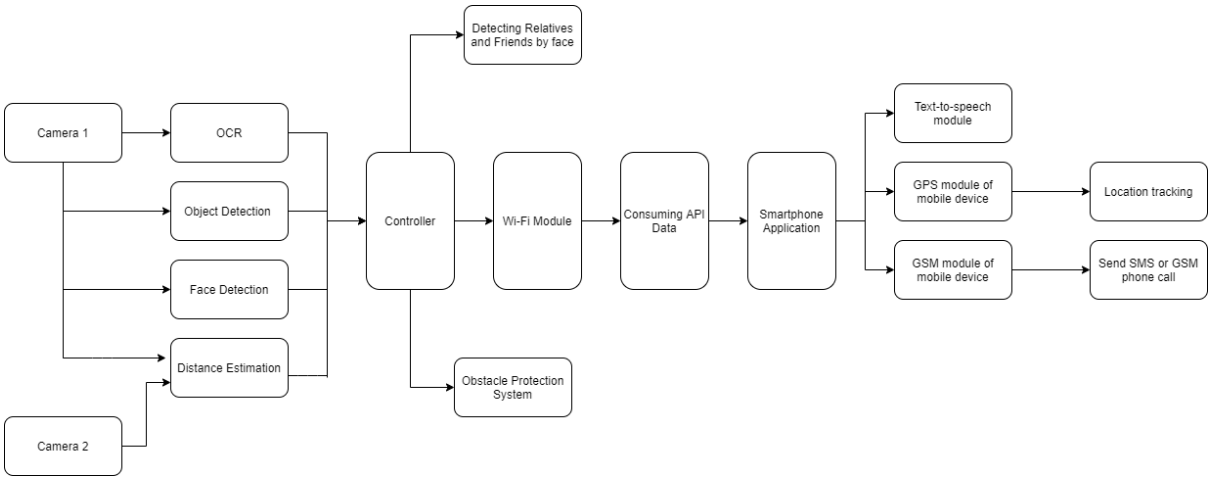


FIGURE 1-13 OUR PROJECT BLOCK DIAGRAM

The next chapter we will be talking about the literature review about this proposed project and an overview of the previously published works on this specific topic

Chapter 2

Literature Review

LITERATURE REVIEW

What is Computer Vision?

Computer Vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, other visual inputs and take actions or make recommendations based on that information. If artificial intelligence (AI) enables computers to think, computer vision enables them to see, observe and understand. Computer vision is one of the hottest fields in the industry right now.

Computer vision refers to a process which enables one to understand images and videos, how they are stored and how they can be manipulated, and data retrieved from them. Computer vision forms the basis for Artificial Intelligence (AI). Today, it has an important role to play in self-driving cars, robotics and even in photo correction apps.

The History of Computer Vision

Scientists and engineers have been trying to develop ways for machines to see and understand visual data for about 60 years. Experimentation began in 1959 when neurophysiologists showed a cat an array of images, attempting to correlate a response in its brain. They discovered that it responded first to hard edges or lines, and scientifically, this meant that image processing starts with simple shapes like straight edges.

At about the same time, the first computer image scanning technology was developed, enabling computers to digitize and acquire images. Another milestone was reached in 1963 when computers were able to transform two-dimensional images into three-dimensional forms. In the 1960s, AI emerged as an academic field of study, and it also marked the beginning of the AI quest to solve the human vision problem.

1974 saw the introduction of optical character recognition (OCR) technology, which could recognize text printed in any font or typeface. Similarly, intelligent character recognition (ICR) could decipher hand-written text using neural networks. Since then, OCR and ICR have found their way into document and invoice processing, vehicle plate recognition, mobile payments, machine translation and other common applications.

In 1982, neuroscientist David Marr established that vision works hierarchically and introduced algorithms for machines to detect edges, corners, curves and similar basic shapes. Concurrently, computer scientist Kunihiko Fukushima developed a network of cells that could recognize patterns. The network, called the Neocognitron, included convolutional layers in a neural network.

By 2000, the focus of study was on object recognition, and by 2001, the first real-time face recognition applications appeared. Standardization of how visual data sets are tagged and annotated emerged through the 2000s. In 2010, the ImageNet data set became available. It

contained millions of tagged images across a thousand object classes and provides a foundation for CNNs and deep learning models used today. In 2012, a team from the University of Toronto entered a CNN into an image recognition contest.
The model, called Alex Net, significantly reduced the error rate for image recognition. After this breakthrough, error rates have fallen to just a few percent.

OPENCV (Open-Source Computer Vision)

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 the commercial products. OpenCV makes it easy for businesses to utilize and modify the code. it is a library used for image processing. 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 user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies. Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups 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.
Originally developed by Intel, OpenCV was later supported by Willow Garage then Itseez, in turn later acquired by Intel. The first OpenCV version was 1.0. Released under a BSD license, this cross-platform library is free for both academic and commercial use under the open-source Apache 2 License. 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 OpenCV 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. the main aim of creating OpenCV was real-time applications for computational efficiency. Since 2011, OpenCV also offers GPU acceleration for real-time operations. Upon integration with other libraries, such as NumPy, Python can process the

OpenCV array structure for analysis. Identifying image patterns and its several features needs use of vector space and carrying out mathematical operations on these features.
In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images.

Application of Open CV

1. **Face detection / face recognition/ facial recognition system:** A facial recognition system is a technology that can match a human face from a digital image or a video frame against a database of faces. It is used to authenticate users through ID verification services and works by pinpointing and measuring facial features from a given image.
2. **Ego motion estimation:** Ego motion refers to the 3D motion of a camera within an environment. In the context of computer vision, ego motion deals with estimating a camera's motion relative to a rigid scene. An example of ego motion estimation would be estimating a car's moving position with respect to lines on the road or street signs being seen from the car itself. The estimation of ego motion is important in autonomous robot navigation applications.
3. **Gesture recognition:** A subdiscipline of computer vision, gesture recognition works with the goal of interpreting human gestures via mathematical algorithms. Gestures can stem from any bodily motion or state but commonly originate from the face or hand.
4. **Human–computer interaction (HCI):** Human-computer interaction (HCI) is a field of research in the design and the use of computer technology, which studies in detail the interfaces between people (users) and computers. HCI researchers focus on the ways in which humans interact with computers and design technologies allowing humans to interact with computers in novel ways.
5. **Mobile robotics:** A mobile robot is a robot that can move in its surroundings. Thus, mobile robotics is a subfield of robotics and information engineering.
6. **Motion understanding:** As the term suggests, motion understanding refers to understanding, interpreting, and overall analyzing the motion of objects in certain environments, such that we are able to better predict their movements and interact better with them.
7. **Object detection:** Object detection is a computer technology related to computer vision and image processing that focuses on identifying instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.
8. **Image segmentation and recognition:** In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.
9. **Stereopsis stereo vision:** Here stereopsis refers to the perception of depth and 3-dimensional structure formed on the basis of visual information gained from two cameras.
10. **Structure from motion (SFM):** Structure from motion (SFM) is a photogrammetric range imaging technique for estimating three-dimensional structures from two-

dimensional image sequences that can be combined with local motion signals. It is studied in the fields of computer vision and visual perception.

11. **Motion tracking:** Also known as video tracking, motion tracking is the process of locating a moving object (or multiple objects) over time using a camera.
12. **Augmented Reality (AR):** Augmented Reality (AR) is basically an interactive experience of a real-world environment where the objects in the real world get enhanced by computer-generated perceptual information, often across several sensory faculties, be it visual, auditory, haptic, somatosensory, and olfactory.

Advantages of using OpenCV

1. Simple to learn, lots of tutorial available.
2. Compatibility with leading coding languages: Works with almost all the famous languages.
3. Free to use (Open Source).

Open-CV can do

1. Read and Write Images.
2. Detection of faces and its features.
3. Detection of shapes like Circle, rectangle, etc. in image.
4. EX. Detection of coin in images.
5. Text recognition in images.
6. EX. Reading Number Plates.
7. Modifying image quality and colors.
8. EX. Instagram, Cam Scanner.
9. Developing Augmented reality apps.
10. Image Processing.
11. Edge Detection and Image Gradients.
12. Dilation, Opening, Closing, and Erosion.
13. Perspective Transformation.
14. Image Pyramids.
15. Cropping.
16. Scaling, Interpolations, and Re-Sizing.
17. Thresholding, Adaptive Thresholding, and Binarization.
18. Sharpening.
19. Blurring.
20. Contours.
21. Line Detection Using Hough Lines.
22. Finding Corners.
23. Counting Circles and Ellipses.
24. Rotate Image.

OpenCV helps use to rotate the image by any degree ranges from 0 to 360 degrees.

What is Image Processing?

Image processing refers to the analysis and manipulation of a digitized image, especially in order to improve its quality. In other words, it involves operations performed on images so as to get their enhanced versions or to glean/ extract some useful information from them. Typically, such operations comprise the steps of importing the image, analyzing, and manipulating it and getting the output that is typically a modified image or a report based on the image analysis.

As a result, image processing helps with tasks such as reading and writing images, detection of faces and its features, detection of shapes such as circles, squares, rectangles in an image (for instance, detection of coins in images), text recognition in images (reading vehicle number plates), changing image quality and colors / filters (think apps like Instagram) and developing Augmented Reality functionality or apps.

Image processing convert images to numbers by pixel values, every number represents the pixel intensity at that particular location. pixel values for a grayscale image where every pixel contains only one value as the intensity of the black color at that location. Note that color images will have multiple values for a single pixel. These values represent the intensity of respective channels – Red, Green, and Blue channels for RGB images.

By default, the imread function reads images in the BGR (Blue-Green-Red) format. We can read images in different formats using extra flags in the imread function:

- cv2.IMREAD_COLOR: Default flag for loading a color image
- cv2.IMREAD_GRAYSCALE: Loads images in grayscale format
- cv2.IMREAD_UNCHANGED: Loads images in their given format, including the alpha channel.
- Alpha channel stores the transparency information – the higher the value of alpha channel, the opaquer is the pixel

Types of Image Processing:

1. Changing Color Spaces

A color space is a protocol for representing colors in a way that makes them easily reproducible. We know that grayscale images have single pixel values and color images contain 3 values for each pixel – the intensities of the red, green, and blue channels.

Most computer vision use cases process images in RGB format.

However, applications like video compression and device independent storage – these are heavily dependent on other color spaces, like the Hue-Saturation-Value or HSV color space. As you understand a RGB image consists of the color intensity of different color channels, the intensity and color information are mixed in RGB color space but in HSV color space the color and intensity information are separated from each other. This makes HSV color space more robust to lighting changes.

OpenCV reads a given image in the BGR format by default. So, you'll need to change the color space of your image from BGR to RGB when reading images using OpenCV.

RGB TO GRAYSCALE

GRAY SCALING

Gray-scaling is a method of converting a 3channel image as RGB, HSV, etc. into a single channel image to shades of grey. The final image varies between complete white and black. The importance of Gray-Scaling includes Dimension reduction (converting 3 channels to a single-channel image), Reduce model complexity, etc.

There are a number of commonly used methods to convert an RGB image to a grayscale image such as **Average Method** and **Weighted Method.**

1. ***Average Method***

The Average method takes the average value of R, G, and B as the grayscale value.

Grayscale = (R + G + B) / 3.

Theoretically, the formula is 100% correct. But when writing code, you may encounter uint8 overflow error — the sum of R, G, and B is greater than 255. To avoid the exception, R, G, and B should be calculated respectively.

Grayscale = R / 3 + G / 3 + B / 3.

The average method is simple but doesn't work as well as expected. The reason being that human eyeballs react differently to RGB. Eyes are most sensitive to green light, less sensitive to red light, and the least sensitive to blue light. Therefore, the three colors should have different weights in the distribution. That brings us to the weighted method.

2. ***The Weighted Method***

The weighted method, also called luminosity method, weighs red, green, and blue according to their wavelengths. The improved formula is as follows:

Grayscale = 0.299R + 0.587G + 0.114B

BINARIZATION: GRAYSCALE TO BLACK/WHITE CONVERSION

Binarization converts a grayscale image to a black/white image. This transformation is useful in detecting blobs and further reduces the computational complexity. The critical task is to find a suitable threshold. There are two main methods:

1. **Local thresholding** — calculates the threshold pixel by pixel.
2. **Global thresholding** — calculates the threshold once for all pixels.

1. ***Local Thresholding Method***

With local thresholding method, a threshold is calculated at each pixel, which depends on some local statistics such as mean, range, and the variance of the pixel neighborhood. The image is divided into several sub-blocks and the distribution of gray-value in each block was analyzed.

2. Global Thresholding Method

The global thresholding method take advantage of the image histogram. The image histogram is a type of statistical graph with grayscale value on the x-axis and the number of pixels for each grayscale on the y-axis.

THRESHOLD TECHNIQUES:

Threshold technique is one of the important techniques in image segmentation. This technique can be expressed as: T=T [x, y, p (x, y), f (x, y] (1) Where: T is the threshold value. x, y are the coordinates of the threshold value point. P (X, Y), f (X, Y) are points the gray level image pixels. Threshold image g (X, Y) can be defined:

$$g(x,y)= \begin{cases} 1 & if \quad f(x,y) > T \\ 0 & if \quad f(x,y) \le T \end{cases} \quad (2)$$

2. Resizing Images

Images can be easily scaled up and down using OpenCV. This operation is useful for training deep learning models when we need to convert images to the model's input shape. Different interpolation and down sampling methods are supported by OpenCV, **which can be used by the following parameters:**

INTER_NEAREST: Nearest neighbor interpolation
INTER_LINEAR: Bilinear interpolation
INTER_AREA: Resampling using pixel area relation
INTER_CUBIC: Bicubic interpolation over 4×4-pixel neighborhood
INTER_LANCZOS4: Lanczos interpolation over 8×8 neighborhood
OpenCV's resize function uses bilinear interpolation by default.

3. Image Rotation

The technique of data augmentation. This method allows us to generate more samples for training our deep learning model. Data augmentation uses the available data samples to produce the new ones, by applying image operations like rotation, scaling, translation, etc. This makes our model robust to changes in input and leads to better generalization.

Rotation is one of the most used and easy to implement data augmentation techniques. As the name suggests, it involves rotating the image at an arbitrary angle and providing it the same label as the original image.

4. Image Translation

Image translation can be used to add shift invariance to the model, as by translation we can change the position of the object in the image give more variety to the model that leads to better generalizability which works in difficult conditions. when the object is not perfectly aligned to the center of the image. This augmentation technique can also help the model correctly classify images with partially visible objects.

5. Simple Image Thresholding

Thresholding is an image segmentation method. It compares pixel values with a threshold value and updates it accordingly. OpenCV supports multiple variations of thresholding. In case of adaptive thresholding, different threshold values are used for different parts of the image. This function gives better results for images with varying lighting conditions – hence the term "adaptive".

Otsu's binarization method finds an optimal threshold value for the whole image. It works well for bimodal images (images with 2 peaks in their histogram).

6. Image Segmentation (Watershed Algorithm)

Image segmentation is the task of classifying every pixel in the image to some class. For example, classifying every pixel as foreground or background. Image segmentation is important for extracting the relevant parts from an image. The watershed algorithm is a classic image segmentation algorithm. It considers the pixel values in an image as topography. For finding the object boundaries, it takes initial markers as input. The algorithm then starts flooding the basin from the markers till the markers meet at the object boundaries.

7. Bitwise Operations

Bitwise operations include AND, OR NOT and XOR. You might remember them from your programming class! In computer vision, these operations are very useful when we have a mask image and want to apply that mask over another image to extract the region of interest.

8. Edge Detection

Edges are the points in an image where the image brightness changes sharply or has discontinuities. Such discontinuities generally correspond to: Discontinuities in depth, Discontinuities in surface orientation, Changes in material properties, Variations in scene illumination.

Edges are very useful features of an image that can be used for different applications like classification of objects in the image and localization. Even deep learning models calculate edge features to extract information about the objects present in image.

Edges are different from contours as they are not related to objects rather, they signify the changes in pixel values of an image. Edge detection can be used for image segmentation and even for image sharpening.

9. Image Contours

A contour is a closed curve of points or line segments that represents the boundaries of an object in the image. Contours are essentially the shapes of objects in an image. Unlike edges, contours are not part of an image. Instead, they are an abstract collection of points and line segments corresponding to the shapes of the object(s) in the image. We can use contours to count the number of objects in an image, categorize objects based on their shapes, or select objects of particular shapes from the image.

10. Scale Invariant Feature Transform (SIFT)

Key points are a concept you should be aware of when working with images. These are basically the points of interest in an image. Key points are analogous to the features of a given image. They are locations that define what is interesting in the image. Key points are important, because no matter how the image is modified (rotation, shrinking, expanding, distortion), we will always find the same key points for the image. Scale Invariant Feature Transform (SIFT) is a very popular key point detection algorithm. It consists of the following steps:

- Scale-space extrema detection.
- Key point localization.
- Orientation assignment.
- Key point descriptor.
- Key point matching.

Features extracted from SIFT can be used for applications like image stitching, object detection, etc.

11. Speeded-Up Robust Features (SURF)

Speeded-Up Robust Features (SURF) is an enhanced version of SIFT. It works much faster and is more robust to image transformations. In SIFT, the scale space is approximated using Laplacian of Gaussian.

Laplacian is a kernel used for calculating the edges in an image. The Laplacian kernel works by approximating a second derivative of the image. Hence, it is very sensitive to noise. We generally apply the Gaussian kernel to the image before Laplacian kernel thus giving it the name Laplacian of Gaussian. In SURF, the Laplacian of Gaussian is calculated using a box filter (kernel). The convolution with box filter can be done in parallel for different scales which is the underlying reason for the enhanced speed of SURF (compared to SIFT).

12. Feature Matching

The features extracted from different images using SIFT or SURF can be matched to find similar objects/patterns present in different images. The OpenCV library supports multiple feature-matching algorithms, like brute force matching, KNN feature matching, among others.

the key points extracted from the original image are matched to key points of its rotated version. This is because the features were extracted using SIFT, which is invariant to such transformations.

13. Face Detection

OpenCV supports haar cascade-based object detection. Haar cascades are machine learning based classifiers that calculate different features like edges, lines, etc. in the image. Then, these classifiers train using multiple positive and negative samples.

Trained classifiers for different objects like faces, eyes etc. are available in the OpenCV.

Optical character recognition or optical character reader (OCR)

OCR is a technology that enables you to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable

data. All a scanner is creating an image or a snapshot of the document that is nothing more than a collection of black and white or color dots, known as a raster image. In order to extract and repurpose data from scanned documents, camera images or images only PDFs, you need an OCR software that would single out letters on the image, put them into words and then, words into sentences, thus enabling you to access and edit the content of the original document.
OCR is a technology that recognizes text within a digital image. It is commonly used to recognize text in scanned documents and images. OCR software can be used to convert a physical paper document, or an image into an accessible electronic version with text (recognize the text and convert the document to an editable text file.)
OCR is very useful and popular method in various applications. Optical Character Recognition is a process by which we convert printed document or scanned page to ASCII character that a computer can recognize. The document image itself can be either machine printed or handwritten, or the combination of two. Computer system equipped with such an OCR system can improve the speed of input operation and decrease some possible human errors. Recognition of printed characters is itself a challenging problem since there is a variation of the same character due to change of fonts or introduction of different types of noises.
Accuracy of OCR can be dependent on text preprocessing and segmentation algorithms. Sometimes it is difficult to retrieve text from the image because of different size, style, orientation, complex background of image etc. Difference in font and sizes makes recognition task difficult if preprocessing, feature extraction and recognition are not robust.
There may be noise pixels that are introduced due to scanning of the image. Besides, same font and size may also have bold face character as well as normal one. Thus, width of the stroke is also a factor that affects recognition. Therefore, a good character recognition approach must eliminate the noise after reading binary image data, smooth the image for better recognition, extract features efficiently, train the system and classify patterns. Till now there is no complete OCR for printed Script which gives 100% success rate.

A Brief History of OCR

- 1929 – Digit recognition machine.
- 1953 – Alphanumeric recognition machine.
- 1965 – US Mail sorting.
- 1965 – British banking system.
- 1976 – Kurzweil reading machine.
- 1985 – Hardware-assisted PC software.
- 1988 – Software-only PC software.
- 1994-2000 – Industry consolidation.

OCR consists of two parts:

1. **Text Detection:**

Where the textual part within the image is determined. This localization of text within the image is important for the second part of OCR.

2. **Text Recognition**

Where the text is extracted from the image. Using these techniques together is how you can extract text from any image.

Design of OCR

Various approaches used for the design of OCR systems are discussed below:

1. **Matrix Matching:**

Matrix Matching converts each character into a pattern within a matrix, and then compares the pattern with an index of known characters. Its recognition is strongest on monotype and uniform single column pages.

2. **Fuzzy Logic:**

Fuzzy logic is a multi-valued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/ white etc. An attempt is made to attribute a more humanlike way of logical thinking in the programming of computers. Fuzzy logic is used when answers do not have a distinct true or false value and there is uncertainly involved.

3. **Feature Extraction:**

This method defines each character by the presence or absence of key features, including height, width, density, loops, lines, stems, and other character traits. Feature extraction is a perfect approach for OCR of magazines, laser print and high-quality images.

4. **Structural Analysis:**

Structural Analysis identifies characters by examining their sub features- shape of the image, sub-vertical and horizontal histograms. Its character repair capability is great for low quality text and newsprints.

5. **Neural Networks:**

This strategy simulates the way the human neural system works. It samples the pixels in each image and matches them to a known index of character pixel patterns. The ability to recognize characters through abstraction is great for faxed documents and damaged text. Neural networks are ideal for specific types of problems, such as processing stock market data or finding trends in graphical patterns.

Proposed OCR System

Following steps have been followed in the design of proposed OCR system:

- Preprocessing.
- Segmentation.
- Feature Extraction.
- Classification.

Preprocessing

In the proposed OCR system, text digitization is done by a flatbed scanner having resolution between 100 and 600 dpi. The digitized images are usually in gray tone, and for a clear document, a simple histogram-based threshold approach is sufficient for converting them to two tone images. The histogram of gray values of the pixels shows two prominent peaks, and a middle gray value located between the peaks is a good choice for threshold.

For salt and pepper noise we generally use median filter. Median filter replaces the value of a pixel by the median of gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median), Median filters provide excellent noise reduction capabilities, with considering less blurring than linear smoothing filters.

Segmentation

Segmentation is one of the most important phases of OCR system. By applying good segmentation techniques, we can increase the performance of OCR. Segmentation subdivides an image into its constituent regions or objects. Basically, in segmentation, we try to extract basic constituent of the script, which are certainly characters. This is needed because our classifier recognizes these characters only.

Segmentation phase is also crucial in contributing to this error due to touching characters, which the classifier cannot properly tackle. Even in good quality documents, some adjacent characters touch each other due to inappropriate scanning resolution.

Classification:

Classification is performed based on the extracted features. The architecture of a neural network determines how a neural network transfers its input into output. This transfer can be viewed as a computation.

Feature Extraction:

Feature extraction is one of the most important steps in developing a classification system. This step describes the various features selected by us for classification of the selected characters.

Types of Optical Character Recognition (OCR):

1. **Tesseract**

The Tesseract OCR engine rose from its 1980s roots as a proprietary C/C++ Hewlett-Packard algorithm to become open-sourced in 2005 under the ongoing patronage of Google, following a decade of neglect.

Considered one of the most accurate OCR frameworks, Tesseract's capabilities were widely lauded in the FOSS community, and its associated software, datasets and secondary modules are now effectively perceived as a collective Google initiative.

Tesseract supports 116 languages by default, though others can be adapted to it. In 2020, the Internet Archive, possibly the largest OCR project of the last twenty years, switched to a Tesseract workflow, and described Tesseract as having made a 'major step forward' in accuracy in the preceding years.

Tesseract 4 added a long short-term memory (LSTM) recurrent neural network (RNN) architecture and automatic language recognition.

2. OCROPUS:

It is an open-source OCR system developed for book capture, OCRopus has been famously leveraged as the OCR engine for Google's ReCaptcha algorithms a free document analysis and optical character recognition (OCR) system released under the Apache License, Version 2.0 with a very modular design through the use of plugins. These plugins allow OCRopus to swap out components easily.

OCRopus is currently developed under the lead of Thomas Breuel from the German Research Centre for Artificial Intelligence in Kaiserslautern, Germany and is sponsored by Google. OCRopus is developed for Linux, however, users have reported success with OCRopus on Mac OS X and an application called TakOCR has been developed that installs OCRopus on Mac OS X and provides a simple droplet interface.

3. Kraken:

There have been a few refugees from the splintered OCRopus project. Among them is Kraken, a CUDA-supported turnkey OCR framework that runs on Linux and OSX and requires a number of external libraries in order to run. It can be installed via PIP or Anaconda and must load recognition models from external sources. Though the project features a public model repository, it currently only contains the generalized English language model and a model for Syriac text.

4. Calamari OCR

Another OCRopus dissident, Python 3-based Calamari OCR is a CLI-only framework also derived from Kraken. It offers a model repository with an accent on historical rather than contemporary textual sources, and where French is the primary alternative language to English.

5. Keras OCR

The Python-based deep learning API Keras offers a convolutional recurrent neural network (CRNN) for text recognition which has been utilized in several modular FOSS repositories, including Simple digit OCR (for tf. keras2.1) and keras-ocr, which is easier to implement into a new framework and leverages the PyTorch Character-Region Awareness for Text detection (CRAFT) text detector.

6. EasyOCR

EasyOCR is a well-maintained repository supporting more than 80 languages, offers a demo- site, and supports all popular script types, including Latin, Cyrillic, Chinese and Arabic. With native PIP-based operation on Linux, EasyOCR runs via PyTorch on Windows, can be implemented via Docker, and supports CUDA.

The next chapter will talk about object detecting and diving deeper and look at various algorithms that can be used for object detection. Also, we will talk about mobile applications and their various strategies for implementations.

Chapter 3

Object Detection

INTRODUCTION OF OBJECT DETECTION

Object detection is a phenomenon in computer vision that involves the detection of various objects in digital images or videos. Some of the objects detected include people, cars, chairs, stones, buildings, and animals.

This phenomenon seeks to answer two basic questions:

1. What is the object? This question seeks to identify the object in a specific image
2. Where is it? This question seeks to establish the exact location of the object within the image.

Object detection consists of various approaches such as fast R-CNN, Retina-Net, and Single- Shot MultiBox Detector (SSD). Although these approaches have solved the challenges of data limitation and modeling in object detection, they are not able to detect objects in a single algorithm run. YOLO algorithm has gained popularity because of its superior performance over the aforementioned object detection techniques.

All of the YOLO models are object detection models. Object detection models are trained to look at an image and search for a subset of object classes. When found, these object classes are enclosed in a bounding box and their class is identified. Object detection models are typically trained and evaluated on the COCO dataset which contains a broad range of 80 object classes. From there, it is assumed that object detection models will generalize to new object detection tasks if they are exposed to new training data. Here is an example of me using YOLOv4 to detect cells in the bloodstream.

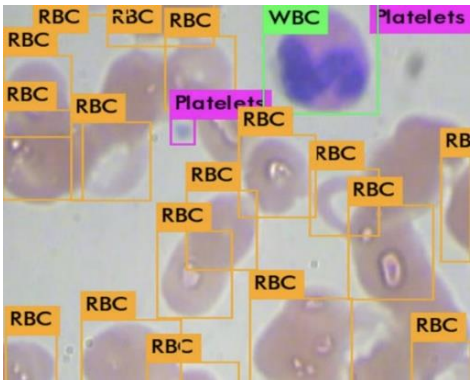


FIGURE 3-1 EXAMPLE OF OBJECT DETECTION

Realtime is particularly important for object detection models that operate on video feeds, such as self-driving cars. The other advantage of real-time object detection models is that they are small and easy to wield by all developers.

Anatomy of Object Detection

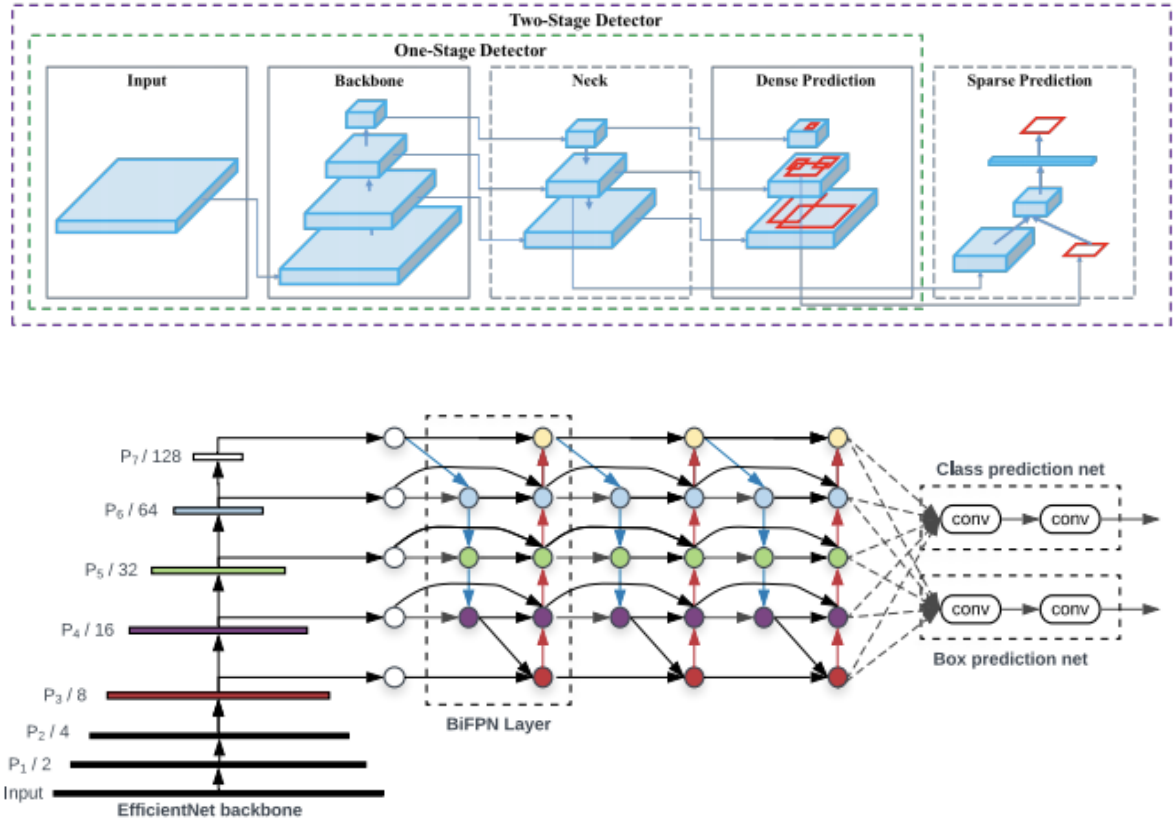


FIGURE 3-2 ANATOMY OF OBJECT DETECTION

All object detectors take an image in for input and compress features down through a convolutional neural network backbone. In image classification, these backbones are the end of the network and prediction can be made of them. In object detection, multiple bounding boxes need to be drawn around images along with classification, so the feature layers of the convolutional backbone need to be mixed and held up considering one another. The combination of backbone feature layers happens in the neck.

It is also useful to split object detectors into two categories: one-stage detectors and two stage detectors. Detection happens in the head. Two-stage detectors decouple the task of object localization and classification for each bounding box. One-stage detectors make the predictions for object localization and classification at the same time. YOLO is a one-stage detector, hence, You Only Look Once.

The First Efficient Face Detector (Viola-Jones Algorithm, 2001)

The Viola–Jones object detection framework is an object detection framework which was proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection.

The problem to be solved is detection of faces in an image. A human can do this easily, but a computer needs precise instructions and constraints. To make the task more manageable,

Viola–Jones requires full view frontal upright faces. Thus, in order to be detected, the entire face must point towards the camera and should not be tilted to either side. While it seems, these constraints could diminish the algorithm's utility somewhat, because the detection step is most often followed by a recognition step, in practice these limits on pose are quite acceptable.

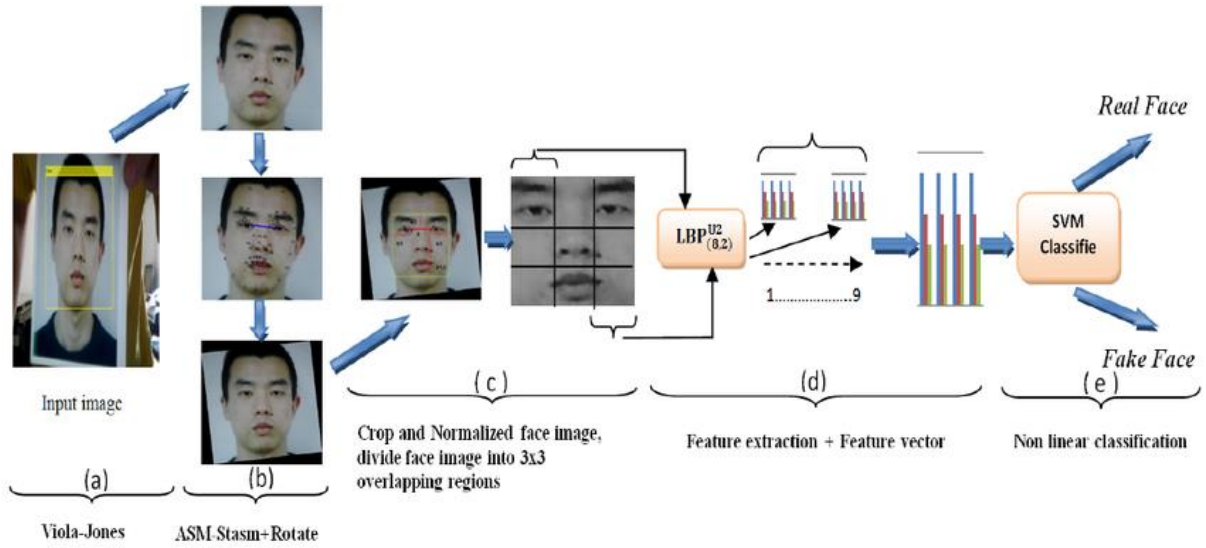


FIGURE 3-3 SHOWS HOW VIOLA-JONES WORKS

The characteristics of Viola–Jones algorithm which make it a good detection algorithm are:

- **Robust** – very high detection rate (true-positive rate) & very low false-positive rate always.
- **Real time** – For practical applications at least 2 frames per second must be processed.
- **Face detection only (not recognition)** - The goal is to distinguish faces from non-faces (detection is the first step in the recognition process).

The algorithm has four stages:

1. Haar Feature Selection.
2. Creating an Integral Image.
3. Adaboost Training.
4. Cascading Classifiers.

Given an image, the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar- like features to detect faces.

YOLOv4 Algorithm

Algorithm: Viola-Jones Face Detection Algorithm

```
1: Input: original test image
2: Output: image with face indicators as rectangles
3: for i ← 1 to num of scales in pyramid of images do
4:     Downsample image to create image_i
5:     Compute integral image, image_ii
6:     for j ← 1 to num of shift steps of sub-window do
7:         for k ← 1 to num of stages in cascade classifier do
8:             for l ← 1 to num of filters of stage k do
9:                 Filter detection sub-window
10:                Accumulate filter outputs
11:            end for
12:            if accumulation fails per-stage threshold then
13:                Reject sub-window as face
14:                Break this k for loop
15:            end if
16:        end for
17:        if sub-window passed all per-stage checks then
18:            Accept this sub-window as a face
19:        end if
20:    end for
21: end for
```

FIGURE 3-4 YOLO ALGORITHM

Much more efficient detection technique (Histograms of Oriented Gradients, 2005)

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is like that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

Robert K. McConnell of Wayland Research Inc. first described the concepts behind HOG without using the term HOG in a patent application in 1986. In 1994 the concepts were used by Mitsubishi Electric Research Laboratories. However, usage only became widespread in 2005 when Navneet Dalal and Bill Triggs, researchers for the French National Institute for Research in Computer Science and Automation (INRIA), presented their supplementary work on HOG descriptors at the Conference on Computer Vision and Pattern Recognition (CVPR). In this work they focused on pedestrian detection in static images, although since then they expanded their tests to include human detection in videos, as well as to a variety of common animals and vehicles in static imagery.

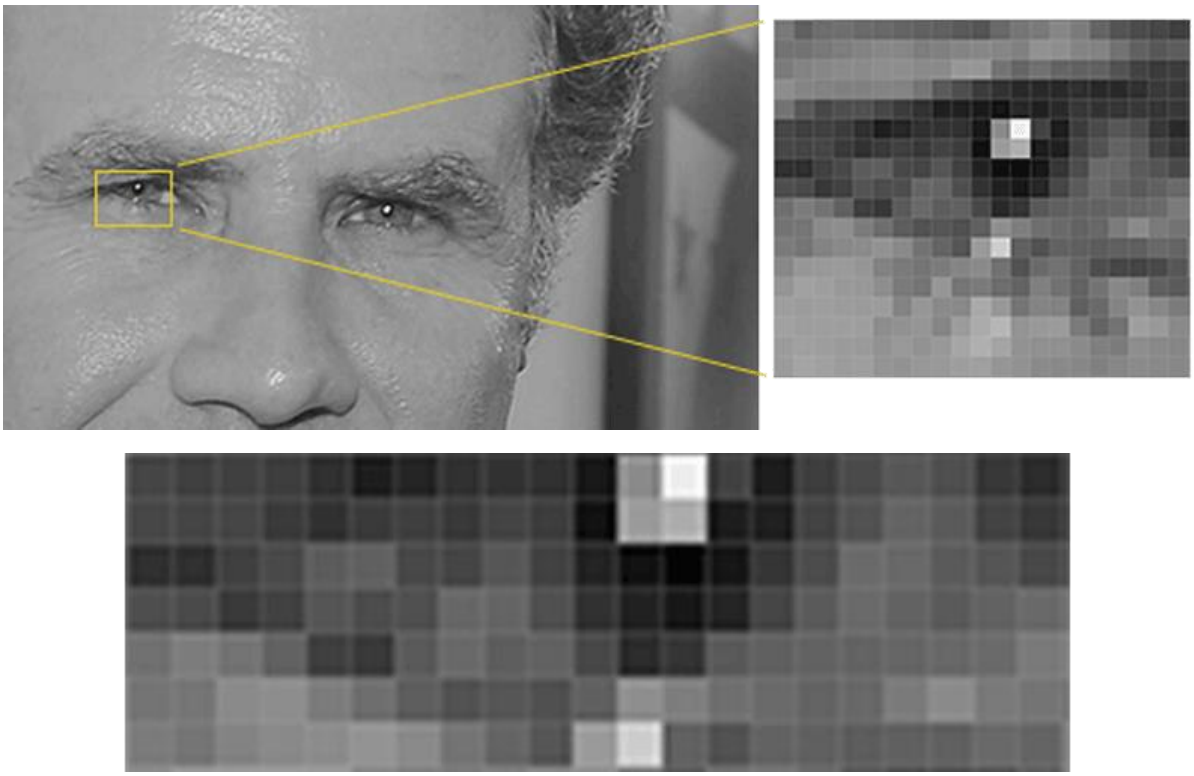


FIGURE 3-5 SHOW HOW HOG ALGORITHM WORKS

THEORY

The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small, connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.

The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions.

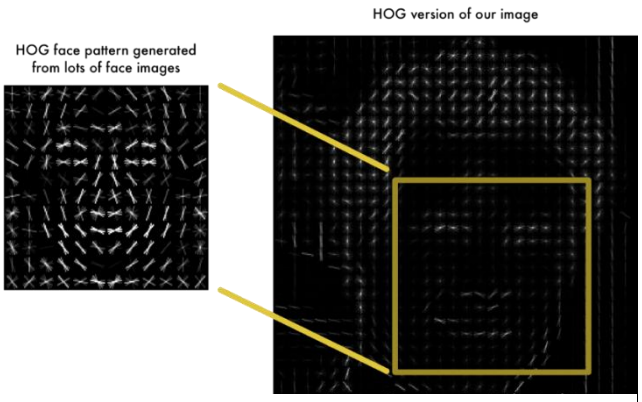


FIGURE 3-6 HOG VERSION OF AN IMAGE

Moreover, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position. The HOG descriptor is thus particularly suited for human detection in images.

THE DEEP LEARNING ERA BEGINS (2012)

Deep learning attempts to mimic the human brain—albeit far from matching its ability— enabling systems to cluster data and make predictions with incredible accuracy. Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

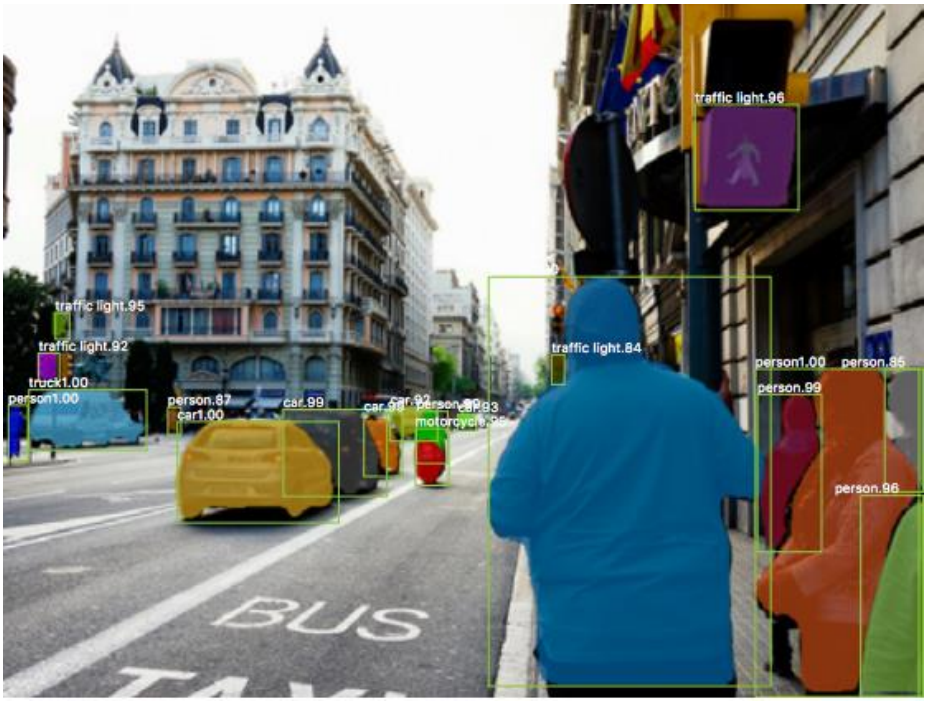


FIGURE 3-7 OBJECT DETECTION USING NEURAL NETWORK

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of

computations through the network are called forward propagation. The input and output layers of a deep neural network are called visible layers.
The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.
Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.
The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex, and there are different types of neural networks to address specific problems or datasets. **For example:**

1. Convolution Neural Networks (CNNs)

Used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.

2. Recurrent neural network (RNNs)

They are typically used in natural language and speech recognition applications as it leverages sequential or times series data.

INTRODUCTION TO FACE DETECTION

A face recognition algorithm is an underlying component of any facial detection and recognition system or software. Specialists divide these algorithms into two central approaches. The geometric approach focuses on distinguishing features. The photo-metric statistical methods are used to extract values from an image. These values are then compared to templates to eliminate variances. The algorithms can also be divided into two more general categories — feature-based and holistic models. The former focuses on facial landmarks and analyzes their spatial parameters and correlation to other features, while holistic methods view the human face as a whole unit.

Eigenfaces

Eigenfaces is a face detection and recognition [2] method that determines face variance in image data sets. It uses these variances to encode and decode faces with machine learning. A set of eigenfaces is a collection of “standardized face ingredients” determined by statistical analysis of many face images. Facial features are assigned mathematical values, as this method doesn’t use digital pictures but rather statistical databases. Any human face is a combination of these values with different percentages.

Fisherfaces

Fisher faces is one of the most popular facial recognition algorithms; it's considered superior to many of its alternatives. As an improvement to the Eigenfaces algorithm, it's often compared to Eigenfaces and considered more successful in class distinction in the training process. The key advantage of this algorithm is its ability to interpolate and extrapolate over lighting and facial expression variation. There are report of 93% accuracy of the Fisher faces algorithm when combined with the PCA method at the preprocessing stage.

Three-Dimensional Recognition

The underlying idea of 3D face recognition technology is the human skull's unique structure. Each person's skull structure is unique and can be described by several dozen parameters. This facial recognition method is based on comparing a 3D facial scan to the database patterns. It has an essential advantage — makeup, facial hair, glasses, and similar factors don't affect the detection and recognition process. The latest [4] has used the technology of mapping the 3D geometry information on a regular 2D grid. It allows the combination of 3D data's descriptiveness with 2D data's computational efficiency and shows the highest performance reported on the FRGC v2 (Face Recognition Grand Challenge 3D facial database).

Thermal Camera

A thermal camera is a device used for monitoring the temperature distribution of the examined surface. The temperature distribution is displayed as a colored picture with different colors corresponding to temperatures. The technology already has several adapting to global changes — smartphone-based immunity certificates, remote fever detection, and thermal facial recognition. Thermal face recognition models are based on the unique temperature patterns of a human face. Human consistent temperature "signatures" are measured with thermal infrared (IR) imaginary. Using the thermal method in face recognition has an undeniable benefit — makeup, facial hair, hats, and glasses don't affect its accuracy. Moreover, it distinguishes twin siblings.

ANFIS

An adaptive neuro-fuzzy interference system (ANFIS) is a type of artificial neural network. This method integrates the principles of neural networks with fuzzy logic principles and combines their advantages in a single structure. ANFIS is used to classify image features extracted from datasets on the preprocessing stage. Data scientists combine this method with a variety of feature extraction algorithms. Thus, some studies incredible 97.1% ANFIS classification accuracy after feature extraction with 2D principal component analysis.

FaceNet

The face recognition system FaceNet, developed by Google researchers in 2015, is based on face recognition benchmark datasets. Available pre-trained models and various open-source third-party implementations make this system quite wide-spread. FaceNet shows excellent in research-conducting surveys, testing performance, and accuracy compared to other algorithms developed

earlier. FaceNet accurately extracts face embeddings, high-quality features used for training face identification systems at later stages.

NEC

The solution developed by the Japanese technology company NEC allows highly accurate identification of people while recognizing age changes. The solution uses Adaptive Region Mixed Matching, a model that focuses on highly similar segments for mapping. The NEC technology divides input and registered images into small segments and focuses only on greater similarity segments. It allows the system to show higher identification accuracy, even in the case of the face wearing a mask or glasses. As its underlying, the NEC solution uses generalized learning vector quantization (GLVQ).

IMPLEMENTATION OF MOBILE APPLICATION

Approaches

1. **Native Apps**: It means that one code create app run in one operating system.
    1.1. Android (supported by Google)
        1.1.1. **Java**: This is the old approach
        1.1.2. **Kotlin**: This is the new approach
    1.2. IOS (Supported by Apple)
        1.2.1. **Swift**: This is the new approach
        1.2.2. **Objective-c**: This is the old approach

2. **Cross-platform apps**: It means that one code create app run in more than one operating system.
    2.1. **Flutter** (*Our approach*): framework supported by google.
    2.2. **React Native**: Supported by Facebook.
    2.3. **Ionic.**
    2.4. **Xamarin**: Supported by Microsoft.

Next Chapter we will talk about the proposed solutions in order to implement and build the proposed projects and the reasons for choosing these implementations.

Chapter 4

Proposed Solution

PROPOSED SOLUTION

The purpose of this project is to help people who have vision problems, regardless of the degree of that problem. It's a complete system to help the blind around with the problems that faces him in his daily, from navigating, to face recognition and object recognition. Therefore, the blind will feel like he is a normal person and hard to be recognized as disabled to avoid the overwhelming feeling of pity and sympathy.

Our proposed system has:

- Text Recognition
- Object Recognition
- Face Recognition
- Distance Measurement
- Mobile app
    - Text to Speech
    - Speech to text
    - Translate

Tesseract

Optical Character Recognition (OCR) system decreases the barrier of the keyboard interface between man and machine to a great extent. And it also helps in office automation with huge saving of time and human effort. Overall, it performs automatic extraction of text from an image, exist in a variety of fonts, and be distorted in all sorts of ways. Tesseract OCR, originally developed at Hewlett Packard from 1984 to 1994, is an open source (under Apache License 2.0) off-line optical character recognition engine. Bristol first started developing Tesseract as a PhD research project in HP Labs. In the year 1995 it was sent to University of Navada, Les Vegas (UNLV), at their it proved its worth against the commercial engines of the time. In the year 2005 it was released by Hewlett Packard and University of Nevada, Las Vegas and presently it is partially funded and maintained by Google. However, output formatting, document layout analysis and graphical user interface is not support by the current version of it. We have used Tesseract version 3.01, which is released in Oct 2011, in the current work.

Tesseract can detect over 100 languages and can process even right-to-left text such as Arabic or Hebrew! No wonder it is used by Google for text detection on mobile devices, in videos, and in Gmail’s image spam detection algorithm.

Tesseract 4.0 has added a new OCR engine that uses a neural network system based on LSTM (Long Short-term Memory), one of the most effective solutions for sequence prediction problems. Although its previous OCR engine using pattern matching is still available as legacy code.

The Tesseract OCR engine was designed from the beginning to be language-independent, but the rest of the engine was developed for English. Its original design goal was that it should recognize white-on-black text. It follows a step-by-step pipeline procedure.
Text of arbitrary length is a sequence of characters, and such problems are solved using RNNs and LSTM is a popular form of RNN.
At the first stage, outlines of the text are gathered by nesting, into Blobs. These blobs are organized into text lines and are broken into words differently according to the kind of character spacing. After that the lines and regions are analyzed for fixed pitch or proportional text. Fixed pitch text is chopped immediately by character cells; however, the proportional text is broken into words using definite spaces and fuzzy spaces. Then the recognition proceeds as a two-pass process.
In the first pass, it attempts recognize each word in turn passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text. Thereafter the second pass is run over the page for the words those were not recognized well enough in the first pass. At last, a final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small cap text. We follow this architecture of Tesseract OCR engine to recognize the Odia characters.

Tesseract OCR Architecture

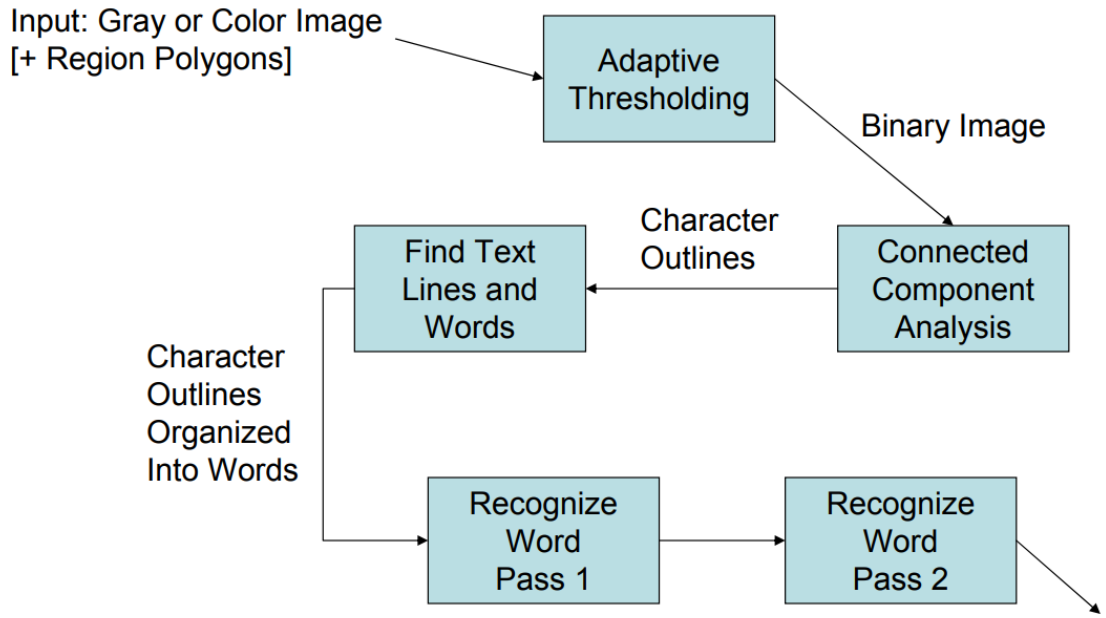


FIGURE 4-1 TESSERACT OCR ARCHITECTURE

Pre-Processing

It is used to enhance the features of the image important for future processing, although geometric transformations of images (EX. rotation, scaling, translation) it suppresses unwilling distortions of image data. Color image does not help in identifying the importance information like edges of the image, so it is importance to convert the RGB image into gray image. In the proposed system binary and Otsu thresholding is used. Thresholding is the simple method to segment the image.

Otsu thresholding automatically finds an optimal threshold based on the observed distribution of pixel value. And to remove the noise in the image fastN1 mean denoising is used this helps to improve the image quality. This preprocessed image is feed to the algorithm for extracting the features and to adjust the weights and biases for the prediction of correct output.

Working of LSTM (Long Short-Term Memory) algorithm

LSTM network is comprised of different memory blocks called cells. These are used to solve the long- term storage problem. Repeating Cell of LSTM has four Neural Network Layers. The Cell state of LSTM allows information to pass through without being changed.

The neural network layer of LSTM consists of sigmoid activation function and a point multiplication. The output of sigmoid activation function varies from 0 to 1.'0' state indicates that not to pass any information and '1' state indicates the complete information to pass through. The below figure shows the block diagram of LSTM network. These networks do not allow any information to be manipulated in the cell state. LSTM cell has three gates Input gate, Forgot gate and output gate.

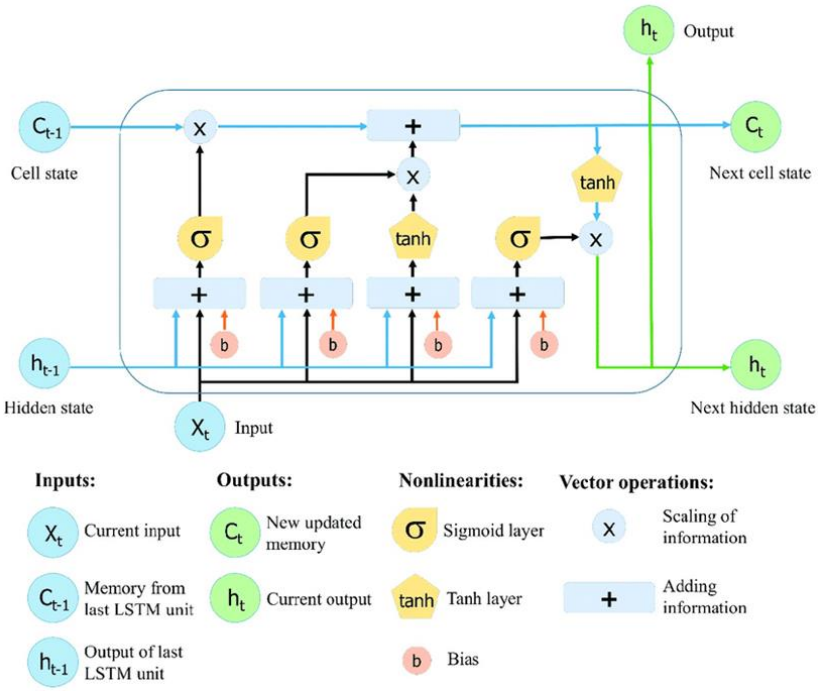


FIGURE 4-2 THE BLOCK DIAGRAM OF LSTM NETWORK

a. Input Gate

The input gate is responsible for the addition of information to the cell state.

- In the Input gate we pass previous output state and present state input into a sigmoid function.
- The sigmoid function squishes the values between 0 and 1. The output of sigmoid function is passed through the tanh function. The tanh function outputs a vector containing all possible values that which squinches outputs from -1 to +1.

- The sigmoid gate output to the tanh function outputs is multiplied and then adding the useful information to the cell state via addition operation.

Once this three-step process is done with, ensure that only that information is added to the cell state is important and is not redundant.

b. Output gate

Output gate is responsible for selecting useful information from the current cell state and showing it out as an output. The functioning of an output gate can again be broken down to three steps:

- In the first step, the output of previous hidden state and current state input is passed into the sigmoid function.
- The new output is multiplied with the tanh function and sigmoid output. The output is new cell state.
- The new cell state and the new hidden state output is passed to the next step.

c. Forgot Gate

As the name indicates the forget gate decides whether the information should be kept or thrown away. It also indicates the proportion of data to be kept. Most widely logistic sigmoid function is used as activation function in LSTM networks. The output of previous hidden state and information from the current input is passed through the activation function. The output of the activation function lies between 0 and 1. If the value is closer to 1 indicates to keep and the value close to zero indicates to forget the information.

Tesseract Configuration

a. Langue (-l):

You can detect a single language or multiple languages with Tesseract as English (by default), Arabic, Chinese, German, French and etc.

b. OCR engine mode (–oem):

Tesseract 4 has both LSTM and Legacy OCR engines. However, there are 4 modes of valid operation modes based on their combination.

Number	OCR engine mode (-oem)
0	Legacy engine only.
1	Neural nets LSTM engine only.
2	Legacy + LSTM engines.
3	Default, based on what is available.

TABLE 4-1 OCR ENGINE MODE

c. Page Segmentation (–psm):

Can be adjusted according to the text in the table for better results

Number	Page Segmentation(-psm)
0	Orientation and script detection (OSD) only.
1	Automatic page segmentation with OSD.
2	Automatic page segmentation, but no OSD, or OCR.
3	Fully automatic page segmentation, but no OSD. (Default)
4	Assume a single column of text of variable sizes.
5	Assume a single uniform block of vertically aligned text.
6	Assume a single uniform block of text.
7	Treat the image as a single text line.
8	Treat the image as a single word.
9	Treat the image as a single word in a circle.
10	Treat the image as a single character.
11	Sparse text. Find as much text as possible in no particular order.
12	Sparse text with OSD.
13	Raw line. Treat the image as a single text line, by passing hacks that are Tesseract-specific.

TABLE 4-2 PAGE SEGMENTATION

Flowchart of Tesseract

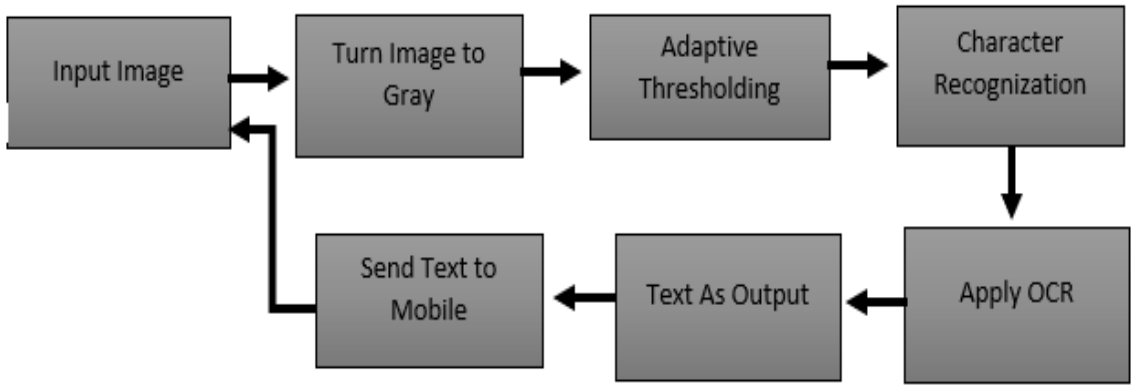


FIGURE 4-3 FLOWCHART OF TESSERACT

Pseudocode of Tesseract:

This program will allow the user to Extract text from images and translate it.

- Capture image from the camera.
- Get the Hight (h) and weight (w) of input image.
- Resize image.
- Convert colorful image to gray image.
- Apply Adaptive Threshold on gray image using Binary threshold.
- Set configuration for tesseract and language.
- Extract data from the image.

```
For number of data !=0 then:
        Split row
            if number of row!=0:
                then Split every column
                  if confidence of OCR >=60:
                      if length of number of column==12:
                      then:
                    get (word number, left, top_weigth, height) from data of image
                            draw rectangular on each recognized word
                            put the recognized text on each box
                            print (recognized text)
        Send recognized text to Mobile
```

INTRODUCTION TO YOLO V4 ALGORITHM

You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a MAP of 57.9% on COCO test-dev. YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.

What is YOLO?

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

The majority of CNN-based object detectors are largely applicable only for recommendation systems. For example, searching for free parking spaces via urban video cameras is executed by slow accurate models, whereas car collision warning is related to fast inaccurate models. Improving the real-time object detector accuracy enables using them not only for hint generating recommendation systems, but also for stand-alone process management and human input reduction. Real-time object detector operation on conventional Graphics Processing Units (GPU) allows their mass usage at an affordable price. The most accurate modern neural networks do not operate in real time and require large number of GPUs for training with a large mini-batch-size. We address such problems through creating a CNN that operates in real-time on a conventional GPU, and for which training requires only one conventional GPU.

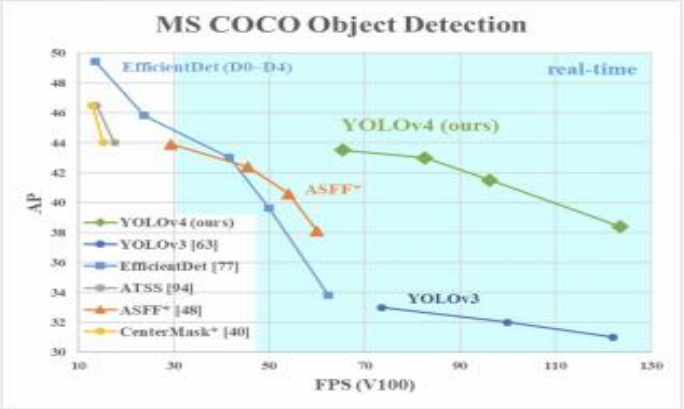


FIGURE 4-4 COCO OBJECT DETECTION

History of YOLOs

The original YOLO (You Only Look Once) was written by Joseph Redmon (now retired from CV) in a custom framework called Darknet. Darknet is a very flexible research framework written in low level languages and has produced a series of the best real-time object detectors in computer vision: YOLO, YOLOv2, YOLOv3, and now, YOLOv4.

The Original YOLO - YOLO was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network.

YOLOv2 - YOLOv2 made several iterative improvements on top of YOLO including BatchNorm, higher resolution, and anchor boxes.

YOLOv3 - YOLOv3 built upon previous models by adding an objectness score to bounding box prediction, added connections to the backbone network layers, and made predictions at three separate levels of granularity to improve performance on smaller objects.

And now onto YOLOv4..

YOLOv4 Backbone Network – Feature Formation

The backbone network for an object detector is typically pretrained on ImageNet classification. Pretraining means that the network's weights have already been adapted to identify relevant features in an image, though they will be tweaked in the new task of object detection.

The authors considered the following backbones for the YOLOv4 object detector

- CSPResNext50
- CSPDarknet53
- EfficientNet-B3

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

TABLE 4-5 PARAMETERS OF NEURAL NETWORKS FOR IMAGE CLASSIFICATION

YOLOv4 Neck – Feature Aggregation

The next step in object detection is to mix and combine the features formed in the ConvNet backbone to prepare for the detection step. YOLOv4 considers a few options for the neck including:

- FPN
- PAN
- NAS-FPN
- BiFPN
- ASFF
- SFAM

The components of the neck typically flow up and down among layers and connect only the few layers at the end of the convolutional network.

YOLOv4 Head – The Detection Step

YOLOv4 deploys the same YOLO head as YOLOv3 for detection with the anchor-based detection steps, and three levels of detection granularity.

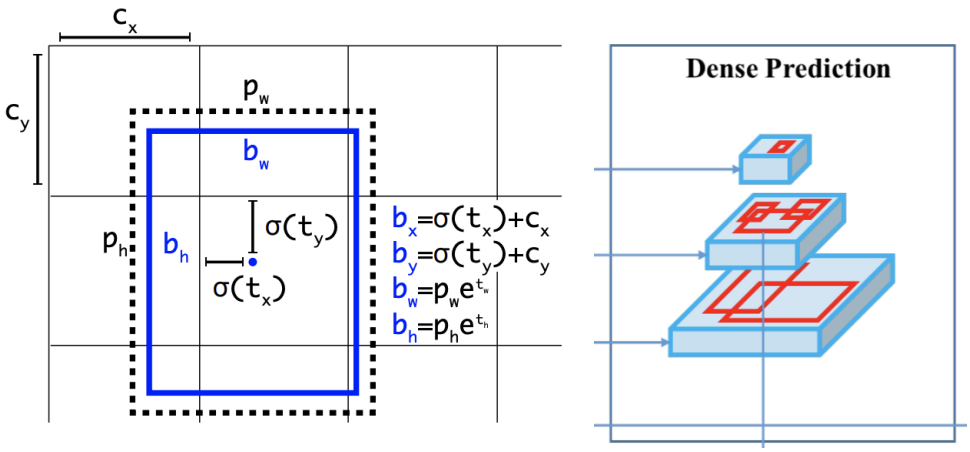


FIGURE 4-6 DETECTION STEP OF YOLO

Why the YOLO algorithm is important

YOLO algorithm is important because of the following reasons:

- **Speed**: This algorithm improves the speed of detection because it can predict objects in real-time.
- **High accuracy**: YOLO is a predictive technique that provides accurate results with minimal background errors.
- **Learning capabilities**: The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

How YOLO Algorithm Works

YOLO algorithm works using the following three techniques:

- Residual blocks.
- Bounding box regression.
- Intersection Over Union (IOU).

Residual Blocks

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.

FIGURE 4-7 SHOWS RESIDUAL BLOCKS

In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

Bounding Box Regression

A bounding box is an outline that highlights an object in an image. Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
- Bounding box center (bx,by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.

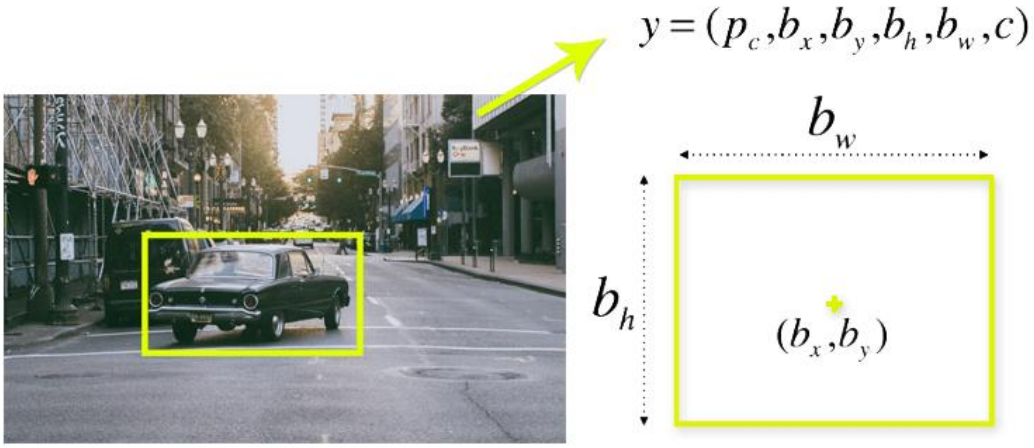


FIGURE 4-8 EXAMPLE OF A BOUNDING BOX

YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

Intersection Over Union (IOU)

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

The following image provides a simple example of how IOU works. There are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.

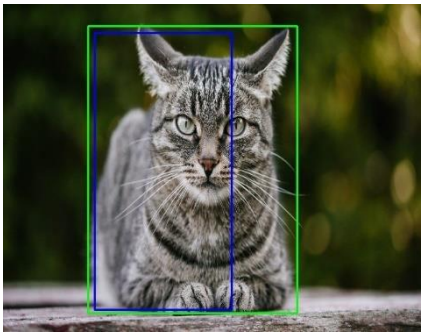


FIGURE 4-9 EXAMPLE OF HOW IOU WORKS

Combination of the three techniques

The following image shows how the three techniques are applied to produce the final detection results.

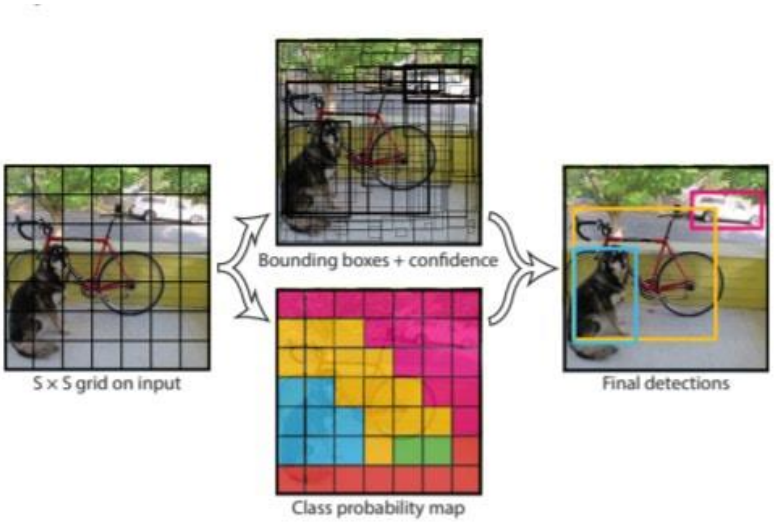


FIGURE 4-10 THE COMBINATION OF RESIDUAL BLOCKS, BOUNDING BOX, AND CLASSIFICATION TECHNIQUES

First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object.

For example, we can notice at least three classes of objects: a car, a dog, and a bicycle. All the predictions are made simultaneously using a single convolutional neural network.

Intersection over union ensures that the predicted bounding boxes are equal to the real boxes of the objects.

This phenomenon eliminates unnecessary bounding boxes that do not meet the characteristics of the objects (like height and width). The final detection will consist of unique bounding boxes that fit the objects perfectly.
For example, the car is surrounded by the pink bounding box while the bicycle is surrounded by the yellow bounding box. The dog has been highlighted using the blue bounding box.

Application of YOLO

YOLO algorithm can be applied in the following fields:

- **Autonomous driving**: YOLO algorithm can be used in autonomous cars to detect objects around cars such as vehicles, people, and parking signals. Object detection in autonomous cars is done to avoid collision since no human driver is controlling the car.
- **Wildlife**: This algorithm is used to detect various types of animals in forests. This type of detection is used by wildlife rangers and journalists to identify animals in videos (both recorded and real-time) and images. Some of the animals that can be detected include giraffes, elephants, and bears.
- **Security**: YOLO can also be used in security systems to enforce security in an area. Let's assume that people have been restricted from passing through a certain area for security reasons. If someone passes through the restricted area, the YOLO algorithm will detect him/her, which will require the security personnel to take further action.

YOLO Flowchart

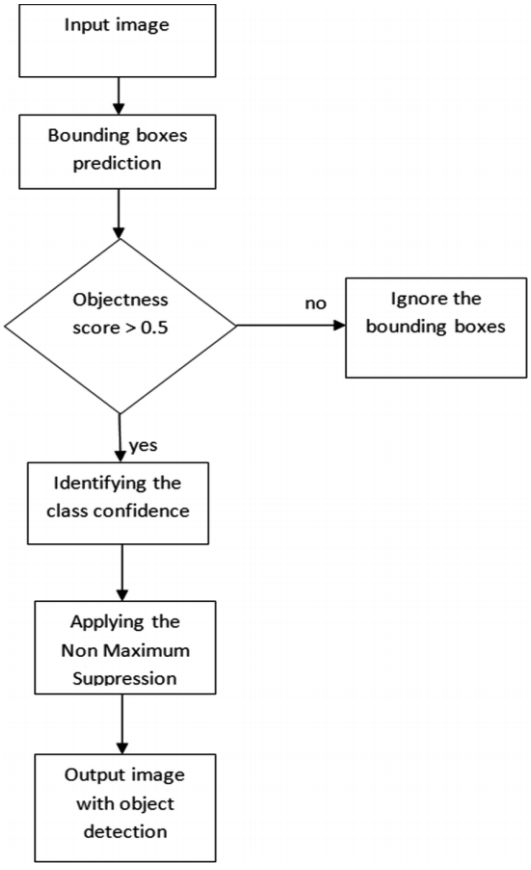


FIGURE 4-11 YOLO FLOWCHART

Face Detection

Have you noticed that Facebook has developed an uncanny ability to recognize your friends in your photographs? In the old days, Facebook used to make you to tag your friends in photos by clicking on them and typing in their name. Now as soon as you upload a photo, Facebook tags everyone for you like magic: Facebook automatically tags people in your photos that you have tagged before.

This technology is called face recognition. Facebook's algorithms can recognize your friends' faces after they have been tagged only a few times. It's pretty amazing technology — Facebook can recognize faces with 98% accuracy which is pretty much as good as humans can do.

Let's learn how modern face recognition works! But just recognizing your friends would be too easy. We can push this tech to the limit to solve a more challenging problem — telling Will Ferrell (famous actor) apart from Chad Smith (famous rock musician)!

How to use Machine Learning on a Very Complicated Problem

We've used machine learning to solve isolated problems that have only one step — estimating the price of a house, generating new data based on existing data and telling if an image contains a certain object. All those problems can be solved by choosing one machine learning algorithm, feeding in data, and getting the result.

But face recognition is really a series of several related problems:

1. First, look at a picture and find all the faces in it.
2. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person.
3. Third, be able to pick out unique features of the face that you can use to tell it apart from other people— like how big the eyes are, how long the face is, etc.
4. Finally, compare the unique features of that face to all the people you already know to determine the person's name.

As a human, your brain is wired to do all of this automatically and instantly. In fact, humans are too good at recognizing faces and end up seeing faces in everyday objects:

Computers are not capable of this kind of high-level generalization (at least not yet…), so we must teach them how to do each step in this process separately.

We need to build a pipeline where we solve each step of face recognition separately and pass the result of the current step to the next step. In other words, we will chain together several machine learning algorithms.

Let's tackle this problem one step at a time. For each step, we'll learn about a different machine learning algorithm. I'm not going to explain every single algorithm completely to keep this from turning into a book, but you'll learn the main ideas behind each one and you'll learn how you can build your own facial recognition system in Python using OpenFace and dlib.

1. Finding all the Faces — Detecting

The first step in our pipeline is face detection. Obviously, we need to locate the faces in a photograph before we can try to tell them apart!

If you've used any camera in the last 10 years, you've probably seen face detection in action:

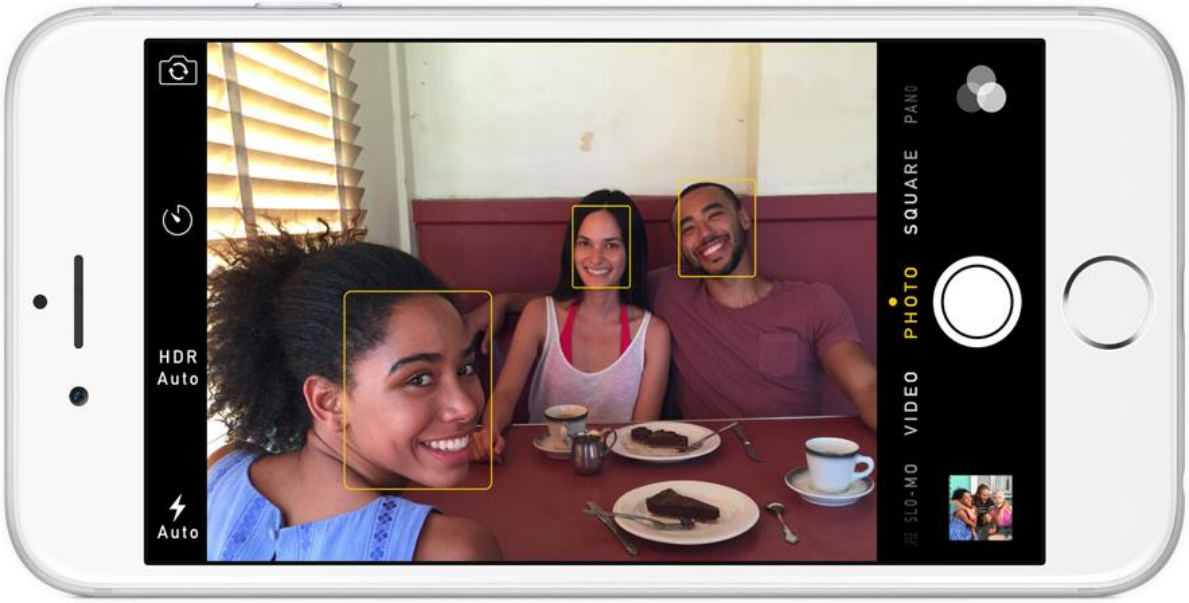


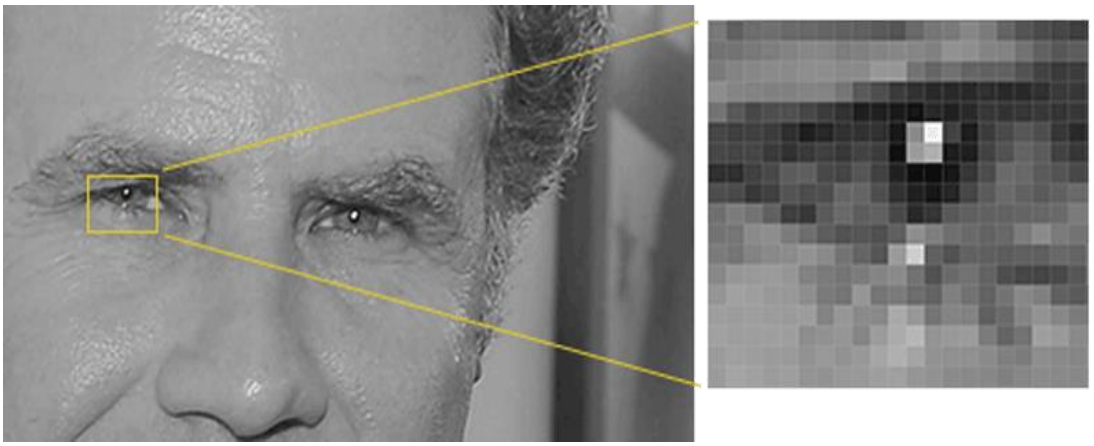
FIGURE 4-12 DETECTING ALL FACES

Face detection is a great feature for cameras. When the camera can automatically pick out faces, it can make sure that all the faces are in focus before it takes the picture. But we'll use it for a different purpose — finding the areas of the image we want to pass on to the next step in our pipeline.

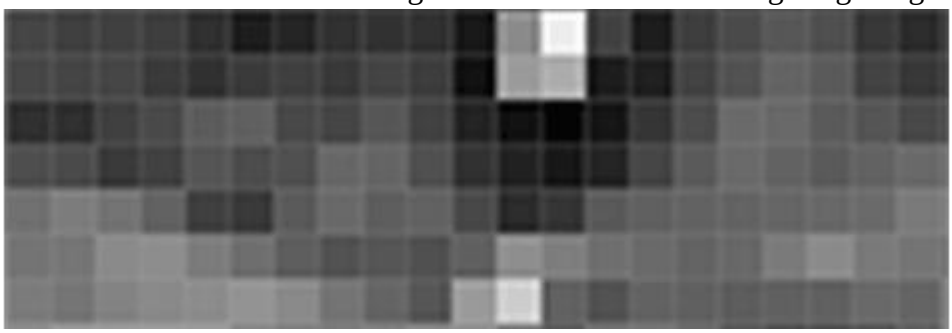
Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. We're going to use a method invented in 2005 called Histogram of Oriented Gradients — or just **HOG** for short.

To find faces in an image, we'll start by making our image black and white because we don't need color data to find faces:

Then we'll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it:



Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:



Looking at just this one pixel and the pixels touching it, the image is getting darker towards the upper right.

If you repeat that process for every single pixel in the image, you end up with every pixel being replaced by an arrow. These arrows are called *gradients* and they show the flow from light to dark across the entire image:

FIGURE 4-13 HOG TECHNIQUE

This might seem like a random thing to do, but there's a good reason for replacing the pixels with gradients. If we analyze pixels directly, dark images and light images of the same person will have totally different pixel values. But by only considering the direction that brightness changes,

both dark images and bright images will end up with the same exact representation. That makes the problem a lot easier to solve!
But saving the gradient for every single pixel gives us way too much detail. We end up missing the forest for the trees. It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image.
To do this, we'll break up the image into small squares of 16x16 pixels each. In each square, we'll count how many gradients point in each major direction (how many points up, point up- right, point right, etc..). Then we'll replace that square in the image with the arrow directions that were the strongest.
The result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way:

FIGURE 4-14 THE ORIGINAL IMAGE BEFORE HOG REPRESENTATION

The original image is turned into a HOG representation that captures the major features of the image regardless of image brightness.
To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces:

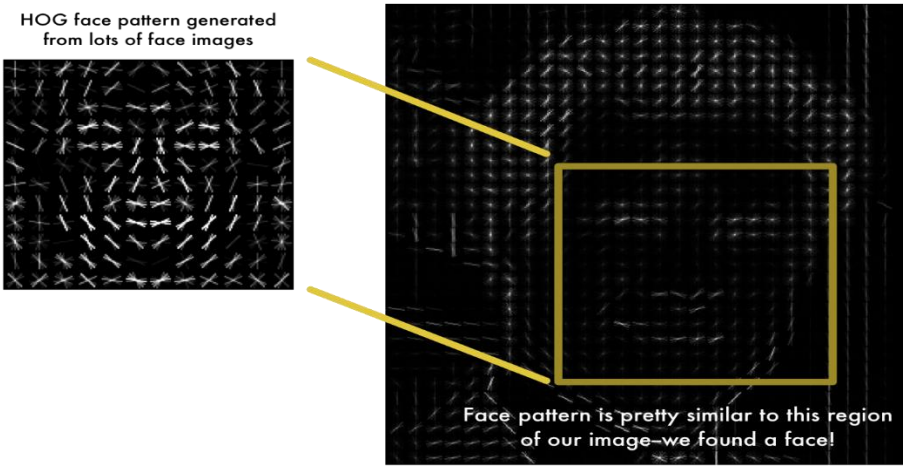


FIGURE 4-15 HOG VERSION OF AN IMAGE

Using this technique, we can now easily find faces in any image:

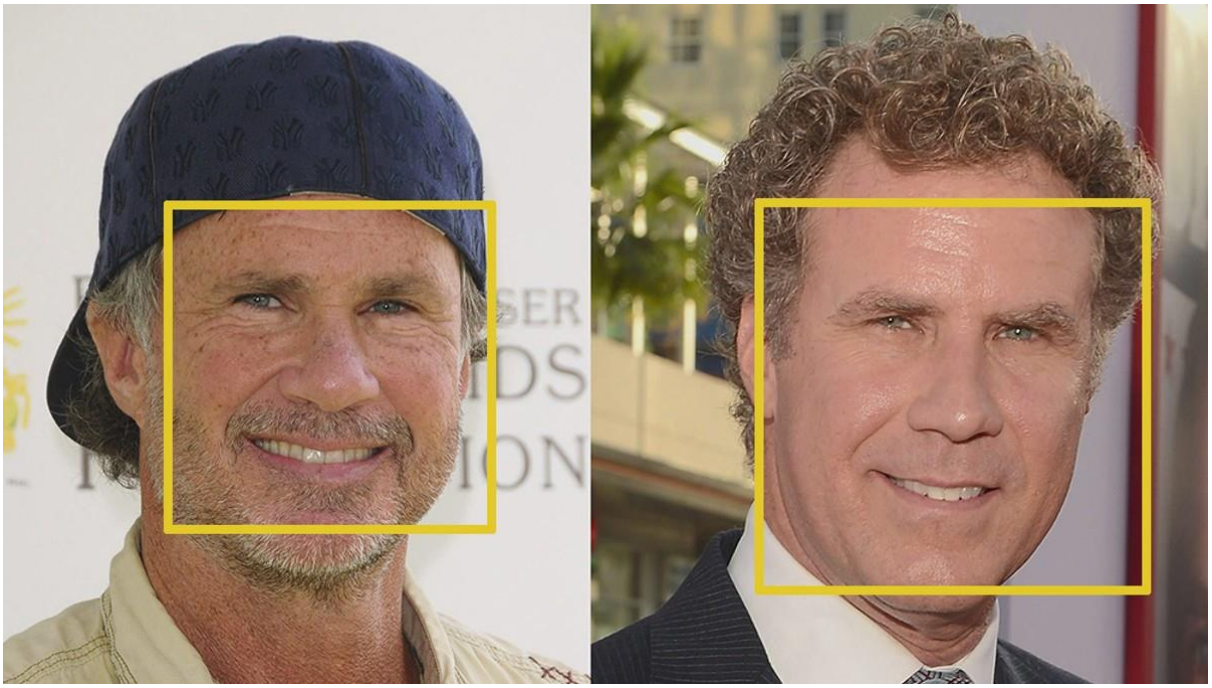


FIGURE 4-16 FACE DETECTION USING HOG REPRESENTATION

2. Posing and Projecting Faces

Whew, we isolated the faces in our image. But now we have to deal with the problem that faces turned different directions look totally different to a computer:

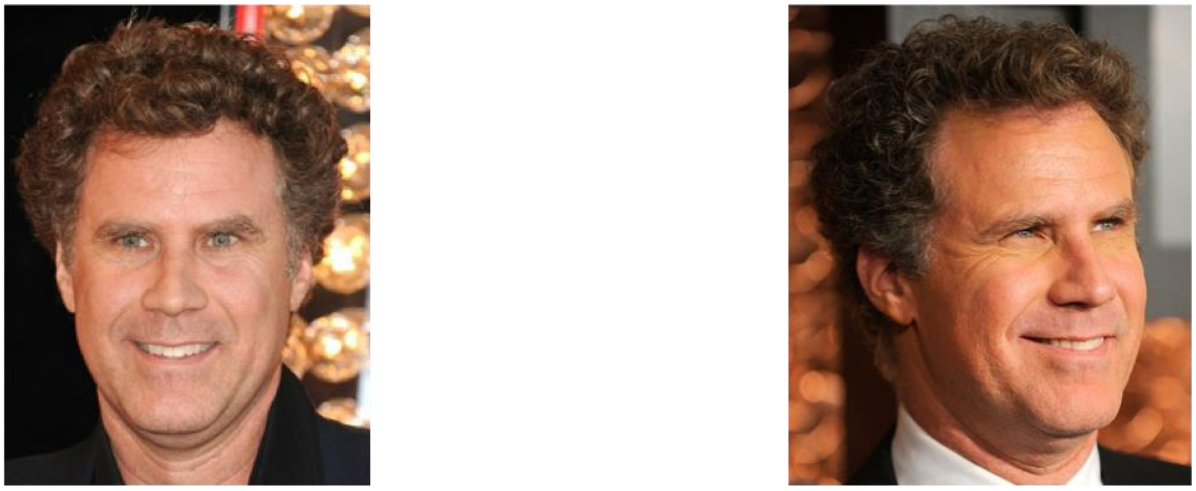


FIGURE 4-17 DIFFERENT POSITIONS OF FACES

Humans can easily recognize that both images are of Will Ferrell, but computers would see these pictures as two completely different people. To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps.

To do this, we are going to use an algorithm called face landmark estimation. There are lots of ways to do this, but we are going to use the approach invented in 2014 by Vahid Kazemi & Josephine Sullivan. The basic idea is we will come up with 68 specific points (called landmarks) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face:

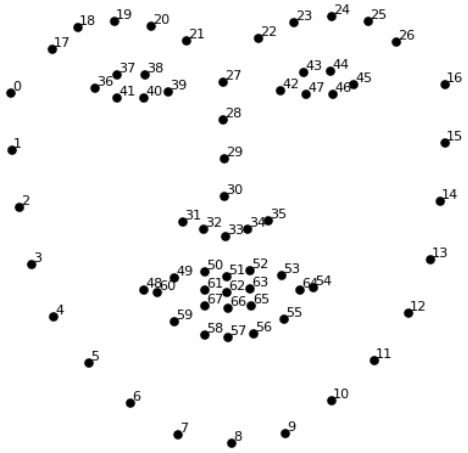


FIGURE 4-18 THE 68 SPECIFIC POINTS ON ANY FACE

The 68 landmarks we will locate on every face. This image was created by Brandon Amos of CMU who works on OpenFace.

Here's the result of locating the 68 face landmarks on our test image:

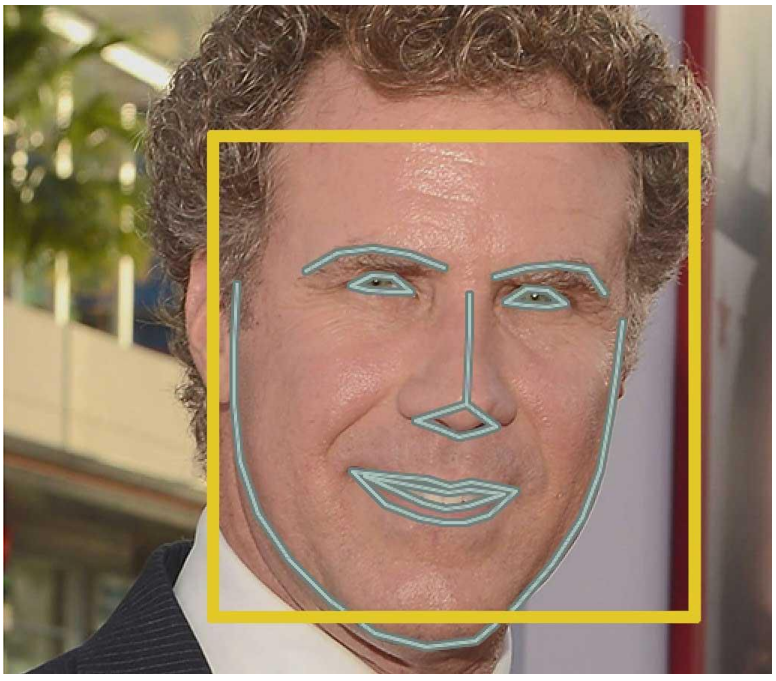


FIGURE 4-19 RESULT OF LOCATING THE 68 FACE LANDMARKS

Now that we know where the eyes and mouth are, we'll simply rotate, scale, and shear the image so that the eyes and mouth are centered as best as possible. We won't do any fancy 3d warps

because that would introduce distortions into the image. We are only going to use basic image transformations like rotation and scale that preserve parallel lines (called affine transformations):

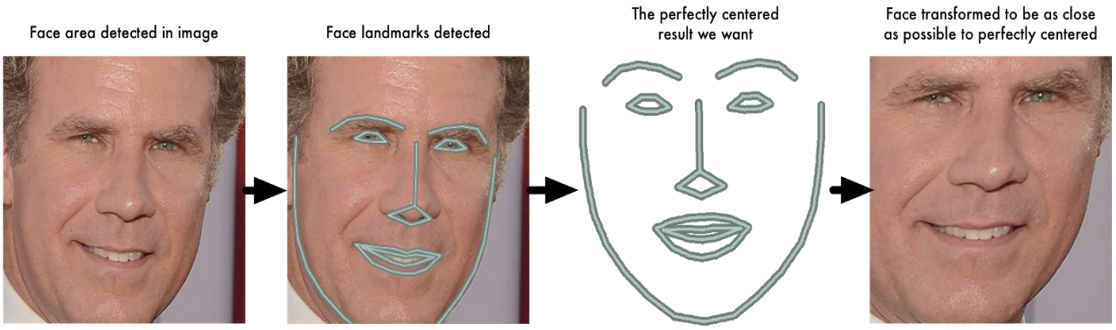


FIGURE 4-20 DETECTING FACE LANDMARKS

Now no matter how the face is turned, we can center the eyes and mouth are in roughly the same position in the image. This will make our next step a lot more accurate.

3. Encoding Faces

Now we are to the meat of the problem — telling faces apart. This is where things get interesting. The simplest approach to face recognition is to directly compare the unknown face we found in Step 2 with all the pictures we have of people that have already been tagged. When we find a previously tagged face that looks very similar to our unknown face, it must be the same person. Seems like a pretty good idea, right?

There's a huge problem with that approach. A site like Facebook with billions of users and a trillion photos can't possibly loop through every previous-tagged face to compare it to every newly uploaded picture. That would take way too long. They need to be able to recognize faces in milliseconds, not hours.

What we need is a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, the spacing between the eyes, the length of the nose, etc. If you've ever watched a bad crime show like CSI, you know what I am talking about:

The most reliable way to measure a face

Ok, so which measurements should we collect from each face to build our known face database? Ear size? Nose length? Eye color? Something else?

It turns out that the measurements that seem obvious to us humans (like eye color) don't really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Network. But instead of training the network to recognize pictures objects like we did last time, we are going to train it to generate 128 measurements for each face.

The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person.
2. Load another picture of the same known person.
3. Load a picture of a totally different person.

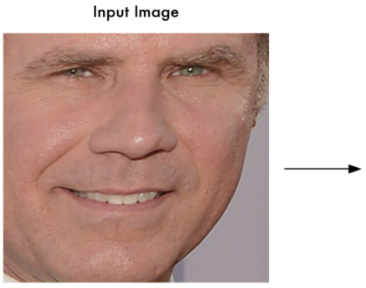
Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart:

After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements.

Machine learning people call the 128 measurements of each face an embedding. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning (especially in language translation). The exact approach for faces we are using was invented in 2015 by researchers at Google but many similar approaches exist.

This process of training a convolutional neural network to output face embeddings requires a lot of data and computer power. Even with an expensive NVidia Telsa video card, it takes about 24 hours of continuous training to get good accuracy.

But once the network has been trained, it can generate measurements for any face, even ones it has never seen before! So, this step only needs to be done once. Lucky for us, the fine folks at OpenFace already did this and they published several trained networks which we can directly use. So, all we need to do ourselves is run our face images through their pre-trained network to get the 128 measurements for each face. Here's the measurements for our test image:



128 Measurements Generated from Image

0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.0198147725313 9	0.10801389068365	-0.0005216327845 1189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.006640171166509 4	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334968328	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266489002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774954907939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

FIGURE 4-21 MEASUREMENTS FOR THE TEST IMAGE

So, what parts of the face are these 128 numbers measuring exactly? It turns out that we have no idea. It doesn't really matter to us. All that we care is that the network generates nearly the same numbers when looking at two different pictures of the same person.

4. Finding the person's name from the encoding

This last step is actually the easiest step in the whole process. All we have to do is find the person in our database of known people who has the closest measurements to our test image.

You can do that by using any basic machine learning classification algorithm. No fancy deep learning tricks are needed. We'll use a simple linear SVM classifier, [8] but lots of classification algorithms could work.

All we need to do is train a classifier that can take in the measurements from a new test image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name of the person!

So, let's try out our system. First, I trained a classifier with the embeddings of about 20 pictures each of Will Ferrell, Chad Smith, and Jimmy Falon:

FIGURE 4-22 EMBEDDING DIFFERENT PICTURES OF FACES

Distance Measurement (Depth Estimation)

Determining the distance from a camera to a marker is a very well-studied problem in the computer vision/image processing space. You can find techniques that are very straightforward and succinct like the triangle similarity. And you can find methods that are complex (albeit, more accurate) using the intrinsic parameters of the camera model.

Two cases can be used to calculate the distance between the camera and an object

1. Using a single camera

To determine the distance from our camera to a known object or marker, we are going to utilize triangle similarity.

The triangle similarity goes something like this: Let's say we have a marker or object with a known width W. We then place this marker some distance D from our camera. We take a picture of our object using our camera and then measure the apparent width in pixels P. This allows us to derive the perceived focal length F of our camera: F = (P x D) / W.

In this case, there is a major issue that we faced while considering using a single camera

We need to find a marker to be a standard point to measure the distance using it, we need to recognize the marker each time and know its dimensions so that we would be able to measure the distance, however, it is not practical for real depth estimation

Using a single camera would require a database for all the objects that are detected so it would be possible to calculate the distance, and that is not a practical solution because not all the objects

have the same dimensions. For example, it's not like all the tables in the real world have the same dimensions.

2. Using Stereo Vision

As humans, we take having two eyes for granted. Even if we lost an eye in an accident we could still see and perceive the world around us. However, with only one eye we lose important information — depth.

Depth perception gives us the perceived distance from where we stand to the object in front of us. To perceive depth, we need two eyes.

Most cameras on the other hand only have one eye, which is why it's so challenging to obtain depth information using a standard 2D camera

You can create a stereo vision system capable of computing depth information using two cameras, such as USB webcams

To Use two cameras to calculate the distance between:

1. Each camera separately using the checkerboard pattern.
2. Calibrate stereo camera setup.
3. Rectification.
4. Triangle Similarity.

1. Camera Calibration

The process of estimating the parameters of a camera is called camera calibration.

This means we have **all the information** (parameters or coefficients) about the camera required to determine an accurate relationship between a **3D point** in the real world and its **corresponding 2D projection (pixel)** in the image captured by that calibrated camera.

Typically, this means recovering two kinds of parameters

1. **Internal parameters** of the camera/lens system. E.g., focal length, optical center, and radial distortion coefficients of the lens.
2. **External parameters**: This refers to the orientation (rotation and translation) of the camera concerning some world coordinate system.

In the image below, the parameters of the lens estimated using geometric calibration were used to un-distort the image.

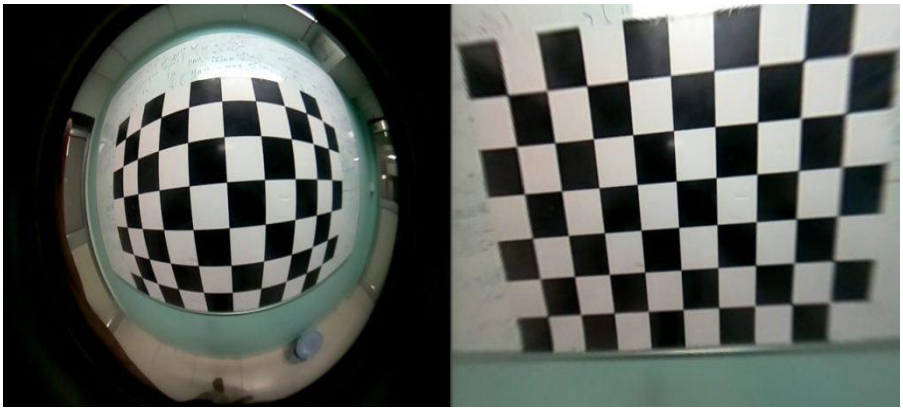


FIGURE 4-23 GEOMETRIC CALIBRATION

To find the projection of a 3D point onto the image plane, we first need to transform the point from the **world coordinate system** to the **camera coordinate system** using the **extrinsic parameters** (Rotation $\mathbf{R}$ and Translation $\mathbf{t}$).

Next, using the **intrinsic parameters** of the camera, we project the point onto the image plane. The equations that relate 3D point (X_w, Y_w, Z_w) in world coordinates to its projection ($\boldsymbol{u}, \boldsymbol{v}$) in the image coordinates are shown below

$$\begin{bmatrix} u' \\ v' \\ z' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$u = \frac{u'}{w'}$$

$$v = \frac{v'}{w'}$$

Where, **P** is a 3×4 Projection matrix consisting of two parts — the **intrinsic matrix** (**K**) that contains the intrinsic parameters and the **extrinsic matrix ([R|t])** that is a combination of a 3×3 rotation matrix **R** and a 3×1 translation/vector.

$$\mathbf{P} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \times \overbrace{\begin{bmatrix} \mathbf{R} \mid \mathbf{t} \end{bmatrix}}^{\text{Extrinsic Matrix}}$$

As mentioned in the previous post, the intrinsic matrix $\mathbf{K}$ is upper triangular.

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where,

f_x,f_y are the x and y focal lengths (yes, they are usually the same).

c_x,c_y are the x and y coordinates of the optical center in the image plane. Using the center of the image is usually a good enough approximation

γ is the skew between the axes. It is usually 0.

The Goal of Camera Calibration

The goal of the calibration process is to find the 3×3 matrix $\mathbf{K}$, the 3×3 rotation matrix $\mathbf{R}$, and the 3×1 translation vector $\mathbf{t}$ using a set of known 3D points (X_w, Y_w, Z_w) and their corresponding image coordinates ($\boldsymbol{u}, \boldsymbol{v}$). When we get the values of intrinsic and extrinsic parameters the camera is said to be calibrated.

In summary, a camera calibration algorithm has the following inputs and outputs

1. **Inputs:** A collection of images with points whose 2D image coordinates and 3D world coordinates are known.
2. **Outputs:** The 3×3 camera intrinsic matrix, the rotation and translation of each image.

Note: In OpenCV, the camera intrinsic matrix does not have the skew parameter. So, the matrix is of the form.

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Different types of camera calibration methods

Following are the major types of camera calibration methods:

1. **Calibration pattern:** When we have complete control over the imaging process, the best way to perform calibration is to capture several images of an object or pattern of known dimensions from different viewpoints. The checkerboard-based method that we will learn in this post belongs to this category. We can also use circular patterns of known dimensions instead of checkerboard patterns.
2. **Geometric clues:** Sometimes we have other geometric clues in the scene like straight lines and vanishing points which can be used for calibration.
3. **Deep Learning-based:** When we have very little control over the imaging setup (e.g. we have a single image of the scene), it may still be possible to obtain calibration information of the camera using a Deep Learning-based method.

Camera Calibration Step by Step

The calibration process is explained by a flowchart given below.

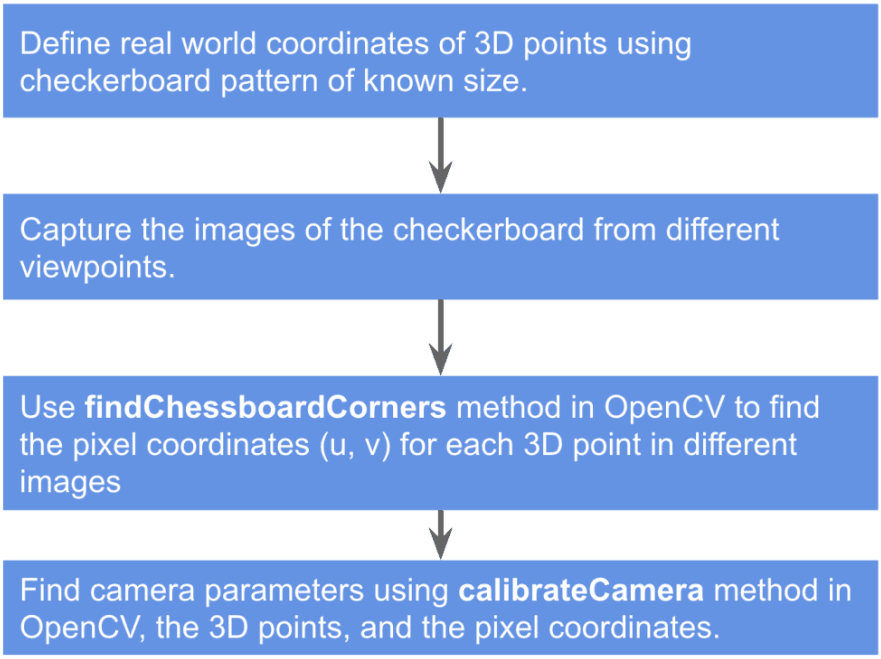


FIGURE 4-24 CAMERA CALIBRATION FLOWCHART

Step 1: Define real-world coordinates with the checkerboard pattern

FIGURE 4-25 DEFINE REAL-WORLD COORDINATES WITH THE CHECKERBOARD PATTERN

World Coordinate System: Our world coordinates are fixed by this checkerboard pattern that is attached to a wall in the room. Our 3D points are the corners of the squares on the checkerboard. Any corner of the above-board can be chosen for the origin of the world coordinate system. The $\mathbf{X_w}$ and $\mathbf{Y_w}$ axes are along the wall, and the $\mathbf{Z_w}$ axis is perpendicular to the wall. All points on the checkerboard are therefore on the XY plane (i.e., $Z_w = 0$).

In the process of calibration, we calculate the camera parameters by a set of known 3D points ($\mathbf{X_w}$, $\mathbf{Y_w}$, and $\mathbf{Z_w}$) and their corresponding pixel location $\boldsymbol{(u, v)}$ in the image.

For the 3D points, we photograph a checkerboard pattern with known dimensions at many different orientations. The world coordinate is attached to the checkerboard and since all the corner points lie on a plane, we can arbitrarily choose $\mathbf{Z_w}$ for every point to be 0. Since points are equally spaced in the checkerboard, the ($\mathbf{X_w}$, $\mathbf{Y_w}$) coordinates of each 3D point are easily defined by taking one point as reference (0, 0) and defining the remaining concerning that reference point.

Why is the checkerboard pattern so widely used in calibration?

Checkerboard patterns are distinct and easy to detect in an image. Not only that, the corners of squares on the checkerboard are ideal for localizing them because they have sharp gradients in two directions. In addition, these corners are also related to the fact that they are at the intersection of checkerboard lines. All these facts are used to robustly locate the corners of the squares in a checkerboard pattern.

Frist camera Calibration (Left Camera):

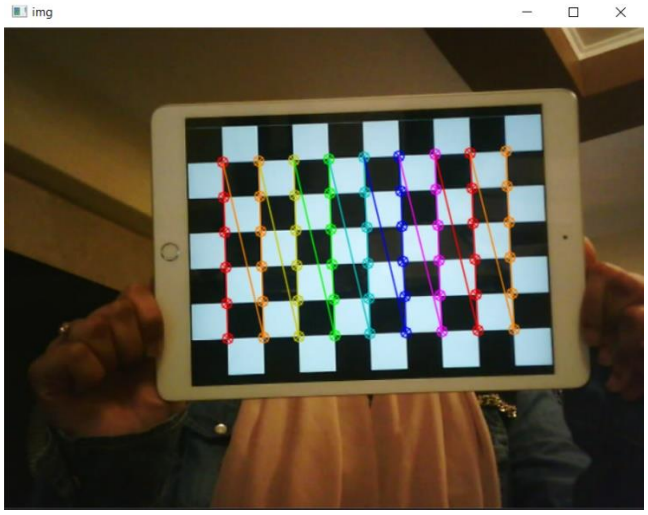


FIGURE 4-26 FIRST CAMERA CALIBRATION

Second camera Calibration (Right Camera):

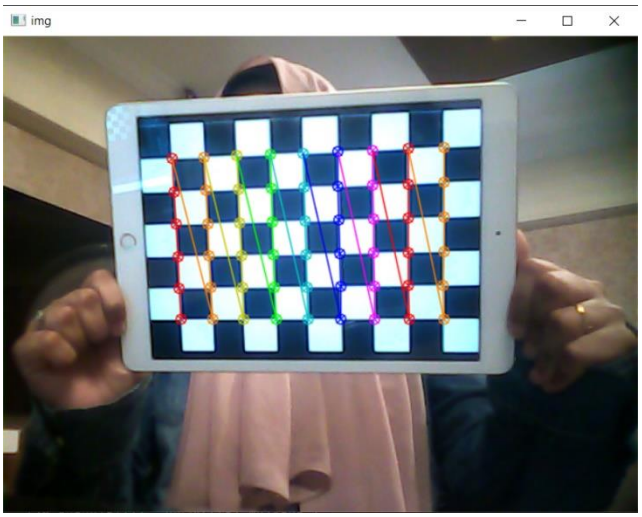


FIGURE 4-27 SECOND CAMERA CALIBRATION

Step 2: Capture multiple images of the checkerboard from different viewpoints

keep the camera constant and photograph the checkerboard pattern at different orientations.

Step 3: Find 2D coordinates of the checkerboard

We now have multiple images of the checkerboard. We also know the 3D location of points on the checkerboard in world coordinates. The last thing we need is the 2D pixel locations of these checkerboard corners in the images.

3.1.Find checkerboard corners

OpenCV provides a built-in function called *find Chessboard Corners* that looks for a checkerboard and returns the coordinates of the corners.

3.2.Refine checkerboard corners

Good calibration is all about precision. To get good results it is important to obtain the location of corners with a sub-pixel level of accuracy.

OpenCV's function *cornerSubPix* takes in the original image, and the location of corners, and looks for the best corner location inside a small neighborhood of the original location. The algorithm is iterative and therefore we need to specify the termination criteria (e.g., number of iterations and/or the accuracy).

Step 4: Calibrate Camera

The final step of calibration is to pass the 3D points in world coordinates and their 2D locations in all images to OpenCV's calibrate camera method. The implementation is based on a paper by Zhengyou Zhang. The math is a bit involved and requires a background in linear algebra.

2. Calibrate stereo camera setup

We now attempt stereo calibration. I assume you have the camera matrices and distortion coefficients of both cameras from the last step. Our first step is to read synchronized frames from both cameras. If you're using the images.

Next, we again find the checkerboard patterns on the two camera frames. If your frames are already synchronized in the previous step, then the image coordinates will work without having to find them again.

Once we have the pixel coordinates of the checkerboard in each image, we simply calibrate the stereo camera setup with a single function call. But we have to tell the function call to keep the camera matrices constant.

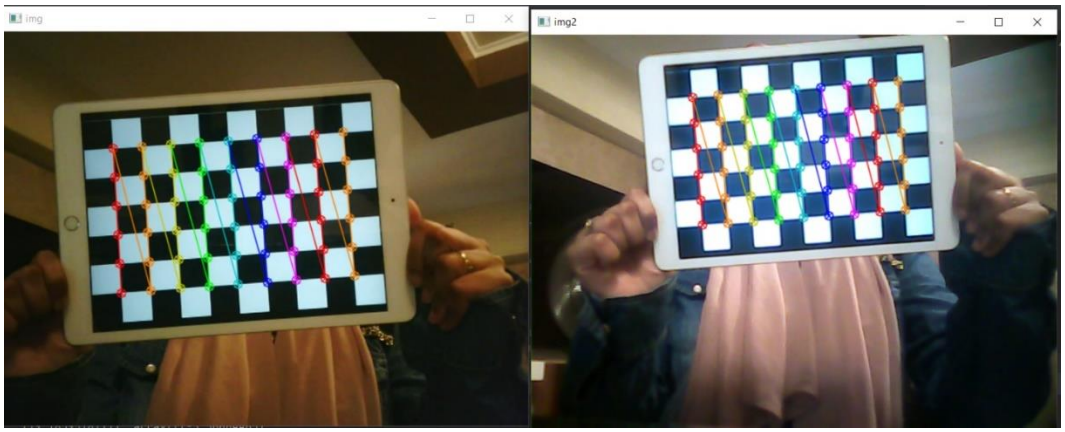


FIGURE 4-28 CALIBRATION OF STEREO CAMERA

The return values of the function call are RMSE, C1 camera matrix (unchanged), C1 distortion coefficients, C2 camera matrix (unchanged), C2 distortion coefficients, rotation matrix, translation vector, essential matrix, and fundamental matrix. The rotation matrix obtained here is the coordinate rotation matrix to go from the C1 coordinate system to the C2 coordinate system. The translation vector is also the location of C2 from C1. So, it does not contain world coordinate rotation and translation vectors, it only provides C2 position and rotation concerning C1. To obtain world coordinate to C2 rotation and translation, calculate as:

$$\mathbf{R}2 = \mathbf{R} * \mathbf{R}1 \text{ and } \mathbf{T}2 = \mathbf{R}T1 + \mathbf{T}$$

One way to get **R**2 and **T**2 is to use the rotation and translation matrices obtained in single-camera calibration from the previous step as **R**1 and **T**1. This means your world coordinate will overlap the bottom left grid of the checkerboard in that frame. If you need absolute world coordinates, you need to determine **R**1 and **T**1 somehow. However, an even simpler choice for **R**1 and **T**1 exists. We simply overlap world coordinates with the coordinates of the first camera. This means **R**1 = eye (3), **T**1 = zeros (3) and **R**2 = **R**, **T**2 = **T**. Therefore, all triangulated 3D points are measured from the C1 camera position in the world.

3. Rectification

Using the camera intrinsics and the rotation and translation between the cameras, we can now apply stereo rectification. Stereo rectification applies rotations to make both camera image planes be in the same plane. Along with the rotation matrices, the stereoRectify method also returns the projection matrices in the new coordinate space.

Image rectification wraps both images. The result is that they appear as if they have been taken only with a horizontal displacement. This simplifies calculating the disparities of each pixel. The depth map algorithm only has the freedom to choose two distinct keyframes from the live camera stream. As such, the stereo rectification needs to be very intelligent in matching & wrapping the images.

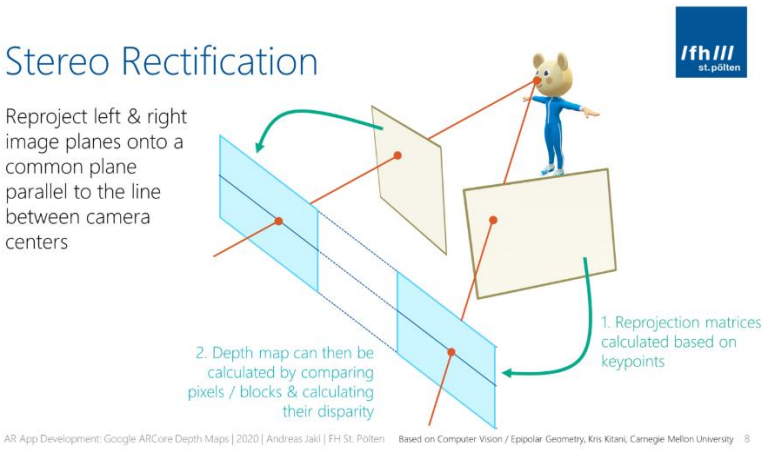


FIGURE 4-29 STEREO RECTIFICATION

In more technical terms, this means that after stereo rectification, all epipolar lines are parallel to the horizontal axis of the image.

To perform stereo rectification, we need to perform two important tasks:

a. **Detect key points** in each image.
b. We then need the best key points where we are sure they are **matched in both images** to calculate reprojection matrices.
c. Using these, we can **rectify the images** to a common image plane. Matching key points are on the same horizontal epipolar line in both images. This enables efficient **pixel/block comparison** to calculate the **disparity map** (= how much offset the same block has between both images) for all regions of the image (not just the key points!).

a) Detecting Key points

To get a feeling for the image, this is what the key points look like from the left image.

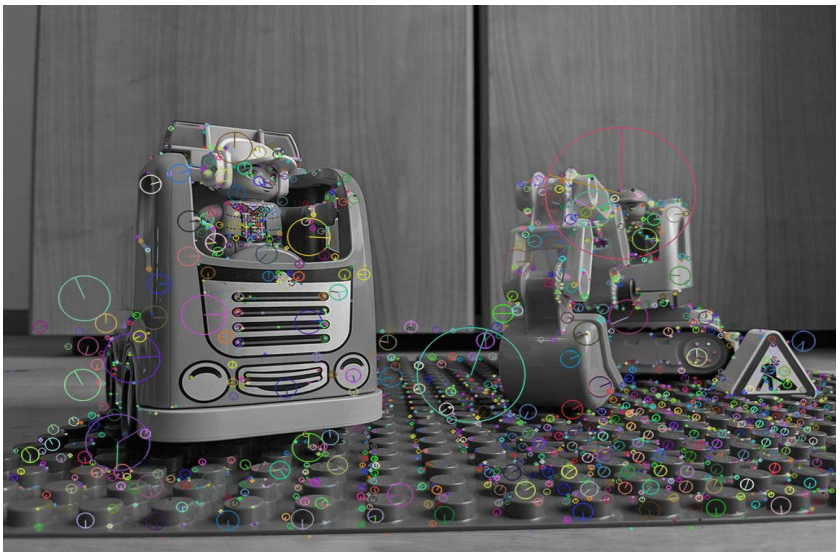


FIGURE 4-30 DETECTING KEY POINTS FROM LEFT IMAGE

b) Matching Key points

There will also be differences in the detected points. As such, to perform stereo rectification, we next need to match the key points between both images. This allows us to detect which are present in both images, as well as their difference in position.

To visually check the matches, we can also draw these. As there are still a lot of matches left

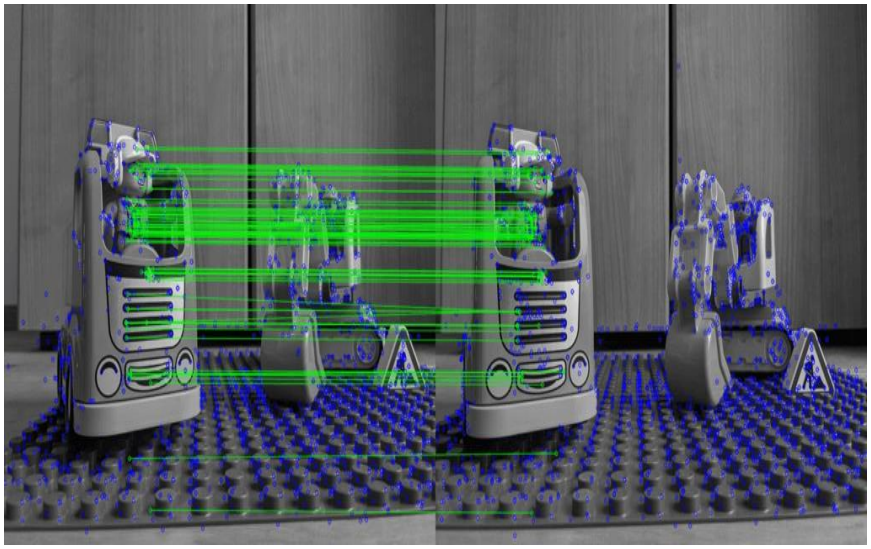


FIGURE 4-31 MATCHING KEY POINTS

c) Stereo Rectification

The matching we have performed so far is a pure 2D keypoint matching. However, to put the images in a 3D relationship, we should use epipolar geometry. A major step to getting there: the fundamental matrix describes the relationship between two images in the same scene. It can be used to map points of one image to lines in another.

In the following example, we observe a cat looking at a mouse. While the cat sees various keypoints of the mouse as points in her field of vision, we "see" the lines between the cat's eyes and the mouse's features as lines.

The origin of the lines we observe is the epipole, and the lines are the epipolar lines.

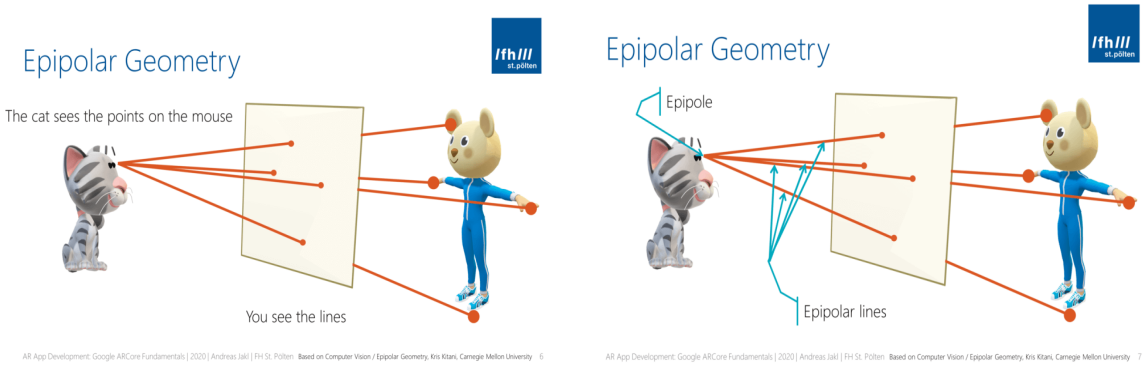


FIGURE 4-32 EPIPOLAR GEOMETRY

4. Triangle Similarity

In the last session, we saw basic concepts like epipolar constraints and other related terms. We also saw that if we have two images of the same scene, we can get depth information from that in an intuitive way. Below is an image and some simple mathematical formulas which prove that intuition.

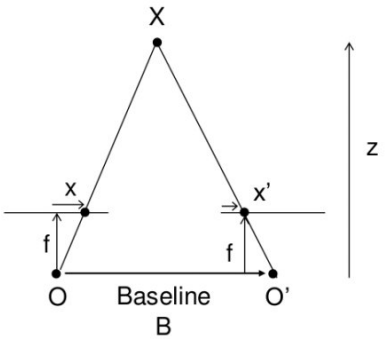


FIGURE 4-33 TRIANGLE SIMILARITY

The above diagram contains equivalent triangles. Writing their equivalent equations will yield the following result:

$$\textbf{Disparity} = x - x' = Bf/Z$$

x and x′ are the distance between points in the image plane corresponding to the scene point 3D and their camera center. B is the distance between two cameras (which we know) and f is the focal length of the camera (already known). So, in short, the above equation says that the depth of a point in a scene is inversely proportional to the difference in distance of corresponding image points and their camera centers. So, with this information, we can derive the depth of all pixels in an image.

The below image contains the original image (left) and its disparity map (right). As you can see, the result is contaminated with a high degree of noise. By adjusting the values of numDisparities and blockSize, you can get a better result.

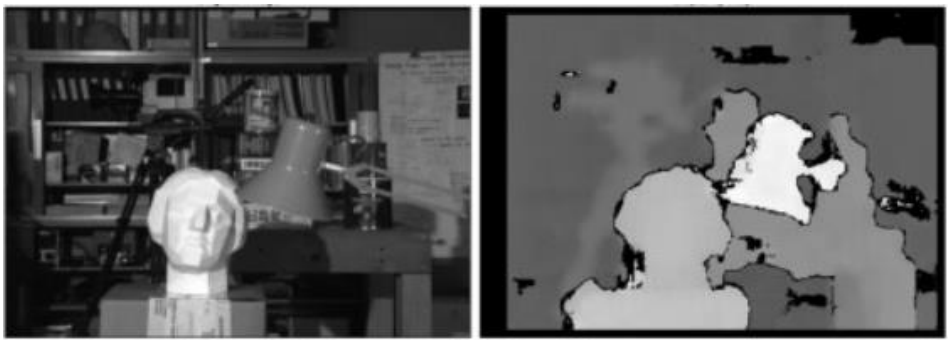


FIGURE 4-34 THE ORIGINAL IMAGE (LEFT) AND ITS DISPARITY MAP (RIGHT)

Application Programming Interface (APIs):

What exactly is an API?

If you want to learn about APIs, you've come to the right place! API stands for application programming interface. APIs are the little pieces of code that make it possible for digital devices, software applications, and data servers to talk with each other, and they're the essential backbone of so many services we now rely on.

Digging deeper, an easy way to understand the definition of an API is to think about the applications that you use every day. In an internet-connected world, web and mobile applications are designed for humans to use, while APIs are designed for other digital systems and applications to use. Websites and APIs both do the same things, like return data, content, images, video, and other information. But APIs don't return all the details that are needed to make things look pretty for the human eye—you only get the raw data and other machine-readable information needed behind the scenes to put the resources being delivered to work, with very little assistance from a human.

What is API integration?

"API integration" is a common Google Search term, and we have good news. The whole reason APIs exist is to support integration. API integration is simply the connection between two (or more) applications, programs, services, or systems, using APIs. Applications use APIs to send and receive data and content between each other. Keep reading for a history of APIs, what they're used for, examples, and more.

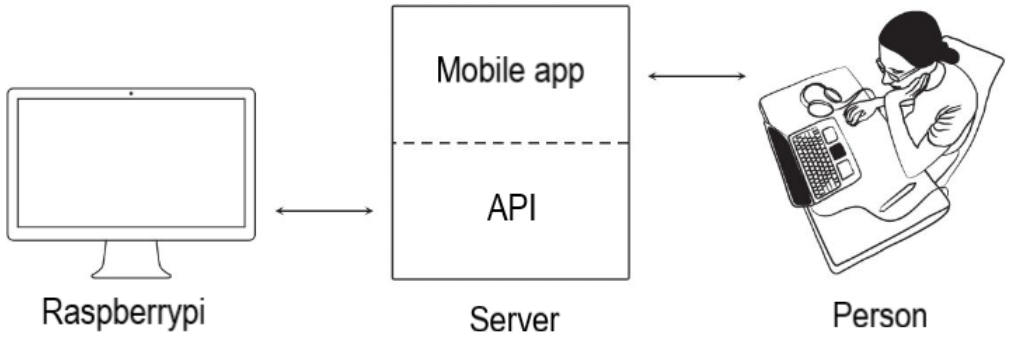


FIGURE 4-35 SHOWS HOW APIS WORK

History of APIs

Web APIs got their start by putting the "commercial" in ".com," powering commerce startups looking to change the way we do business on the web. They took advantage of this new medium to make products and services available to customers via a single website, and as they worked with partners, they sought to automate much of the commerce that was powering the web and included juggernauts like Salesforce, eBay, and Amazon. In 2004, a shift in the API landscape began to emerge as a new breed of API providers started to pop up, offering ways to share information with local and global social networks, led by the likes of Facebook and Twitter. With a strong start in commercial and social applications, APIs continued to grow as everything moved to the cloud, became much more mobile, and provided the foundation for next-generation devices.

What are APIs used for?

What are APIs used for? Lots and lots and lots of things, including:

- APIs power desktop applications.
- APIs are behind most web applications.
- APIs make mobile applications possible.
- APIs are the integrations for no code solutions.
- APIs connect devices to the internet.
- APIs define the networks—or the information passed between applications, systems, and devices.
- APIs even connect everyday things like automobiles, doorbells, dishwashers, and wearable devices.

Why should you care about APIs?

Curious about why you should care about APIs? Here's a very brief list:

- APIs help you access the data you need to get your work done and do daily tasks—whether you're a business user, a student, or using an application just for fun.
- APIs make it possible to integrate different systems together, like Customer Relationship Management systems, databases, or even school learning management systems.
- APIs help different departments, teams, and groups become more agile.
- APIs help organizations, schools, government agencies, and nonprofits strengthen relationships with other organizations, research institutes, and agencies.

How do APIs work?

APIs work by sharing data and information between applications, systems, and devices—making it possible for these things to talk with each other.

Sometimes the easiest way to think about APIs is to think about a metaphor, and a common scenario that a lot of folks' use is that of the customer, a waiter, and a restaurant kitchen: A customer talks to the waiter and tells the waiter what she wants. The waiter takes down the order and communicates it to the kitchen. The kitchen does their work, creating the food, and then the waiter delivers the order back to the customer.

In this metaphor, a customer is like a user, who tells the waiter what she wants. The waiter is like an API, receiving the customer's order and translating the order into easy-to-follow instructions that the kitchen then uses to fulfill that order—often following a specific set of codes, or input, that the kitchen easily recognizes. The kitchen is like a server that does the work of creating the order in the manner the customer wants it, hopefully! When the food is ready, the waiter picks up the order and delivers it to the customer.

Why use APIs? Reasons to use APIs

What are the reasons to use APIs? Almost too many to count, but to truly answer this question, we asked thousands of developers and professionals why they choose to produce or consume APIs in our annual State of the API report. Here are their top answers:

1. ***Integration with internal and external systems:*** One of the top reasons developers use APIs is to integrate one system with another system. For example, if you want your Customer

Relationship Management (CRM) system to integrate with your Marketing Automation system, you could use an API to enable the two systems to talk to one another so that you automatically send a marketing email when a sales representative adds a new prospective customer to the CRM.

2. ***Adding or enhancing functionality of internal and external systems:*** Another common reason that developers use APIs is to add or enhance functionality of internal systems as well as external systems. For example, if you have an internal system that tracks vacation days for managers, you might use an API to allow employees to request days off from their email.
3. ***Adding or enhancing functionality for customers:*** Sometimes when you add or enhance functionality, it's about improving customers' experiences and helping them interact better with your organization. For example, if you worked for a food delivery company, you might use an API to automatically notify customers when their meal is approaching their house.
4. ***Speeding up software and system development:*** APIs allow developers to code and deliver functionality as microservices, instead of big, monolithic applications. By breaking this functionality up, developers can actually speed up software development and system development by eliminating dependencies and reducing the overhead involved in code reviews, testing, and more. Another way that APIs help accelerate development is by allowing frontend and backend teams work in parallel. In other words, a frontend developer can work on creating the frontend of a system, what users and customers see, while a backend developer can work on the underlying system, or what users and customers do not see.
5. ***Reducing operating costs:*** Another reason developers use APIs is to reduce operating costs. APIs can help perform a number of functions that may have been done by humans from pulling reports to sending emails to abstracting data from one system to share with another system. APIs can help reduce operating costs in many, many other ways. Examples include, automatically starting up and shutting down manufacturing systems, creating schedules for workers, or even reducing the number of people who need software licenses.
6. ***Reducing software development costs:*** One of the biggest ways APIs can reduce software development costs is by allowing developers to build reusable components. For example, a backend developer can create a system that serves up information about customers, including their names, email addresses, recent product purchases, etc. Then other developers across the organization can use APIs to grab that information and track payments for finance and accounts payable, help customer service resolve problems faster, or even create recommendations for marketing campaigns. APIs can also help reduce software development costs in a number of other ways such as reducing architectural complexity and reducing the effort required to make changes to systems.
7. ***Improving software and system testing:*** APIs can help improve software and system testing by allowing quality engineering teams to separate tests for frontend components, the parts of software that users see, from tests for backend components, the parts of software that users don't see. API health, quality, and performance can also be checked using automated testing, and API testing can be integrated into the CI / CD pipeline.

8. ***Improving organizational security and governance:*** APIs can be used to improve organizational security. For example, APIs are often used to power Single Sign-On, which is the ability for users to use one username and password to login into multiple systems. This helps avoid the dreaded pile of sticky notes with usernames and passwords, which can be a big security risk. APIs are also often a big part of corporate governance, too. APIs can be used to enforce and automate corporate rules and policies like requiring approval before expenses are paid to employees.
9. ***Enabling mobile applications:*** A lot of mobile applications rely on APIs to deliver important information to mobile users. For example, if you use your mobile phone to check in for a flight and select your seat, APIs can communicate the seat you selected so that flight attendants know where you are seated when you get onboard.
10. ***Reducing outages and non-performing systems****:* Finally, APIs can help reduce outages and non-performing systems. For example, a company might use an API to quickly identify a specific problem with a manufacturing line, and even recommend a fix, which helps maintenance staff fix the system and get back online faster.

The different kinds of APIs

There are many different types of APIs, and many ways to categorize them. Here are some of the most common.

Internal vs. External vs. Partner APIs

One way to categorize APIs is by who has access to them:

- Internal APIs are APIs that are private and only used by your team, department, company, or organization.
- External APIs, also known as public APIs or open APIs (which is not to be confused with OpenAPI), are publicly available APIs that are available for anyone to use.
- Partner APIs is private and shared only with specific, integration partners outside of your organization.

API architectural styles

When it comes to API architecture, there are a number of styles—some newer, some older—and all have a place in the API ecosystem. Defining "architectural styles" broadly, here is a list of the most popular styles listed in order of how frequently they're used:

1. ***REST API:*** REST is an acronym for REpresentational State Transfer. REST APIs rely on a few guiding principles such as a client-server structure, simple, uniform interfaces to communicate across systems, stateless operations, and more.
2. ***Webhooks:*** Webhooks are event-based, and simply put, are automated messages sent from one system to another system anytime an event occurs. Webhooks are even referred to 'reverse APIs' as a concept to check for changes in data.
3. ***SOAP API*****:** SOAP is an acronym for Simple Object Access Protocol. SOAP APIs are more structured and formalized than other APIs, they are reliable and trusted, but can be slower than other APIs. SOAP APIs uses an XML-based messaging protocol which includes the Envelope, Header, and Body tags as required by the endpoint.

4. ***GraphQL API:*** GraphQL is an acronym for Graph Query Language. The Graph Query Language defines how one API asks another API for information and instead of relying on how the server defines the endpoint, a GraphQL query can ask for a specific piece of information. GraphQL was originally created by Facebook as an internal tool in 2012 but they publically released it in 2015 as an open-source language for APIs.
5. ***WebSocket API:*** WebSocket APIs rely on the WebSocket computer communications protocol, which is a full-duplex communication channel over a single TCP connection. Compare WebSocket protocol to HTTP (HyperText Transfer Protocol), which is a half-duplex communication. WebSocket APIs provide a standard way for servers to send information and data to clients, even when the client is not requesting data. WebSocket APIs also allow data to be communicated between clients and servers, while keeping connections open.
6. ***gRPC API:*** The RPC in gRPC stands for Remote Procedure Call; gRPC APIs were originated by Google. In gRPC, a client can call on a server just like it is a local object, making it easier for distributed applications and systems to communicate with one another.
7. ***Server-sent event API:*** Server-Sent Events, also known as SSE, is a technology that relies on data being pushed from the server. This allows a client to receive updates automatically via a HTTP connection.
8. ***AMQP API:*** AMQP is an abbreviation for Advanced Message Queuing Protocol. AMQP is a protocol that follows open standards and works at the application layer. AMQP is best suited for message-oriented middleware, and like other protocols, AMQP dictates how messaging providers and clients communicate with each other. (Hint: message is in the name!) There are a few features that distinguish AMQP including its ability to queue and route messages, which support the reliability and security of AMQP.
9. ***MQTT APIs:*** MQTT is an abbreviation for Message Queuing Telemetry Transport. The MQTT messaging protocol is defined by Organization for the Advancement of Structured Information Standards, better known as OASIS. MQTT is well-suited for the Internet of Things (IoT), in part because it is extremely lightweight. MQTT allows devices to publish and/or subscribe to messages.
10. ***EDI:*** EDI is an abbreviation for Electronic Data Interchange, and it's been around for a long time, since the '70s! The idea behind EDI is to allow businesses to communicate electronically with each, typically transmitting information that was written on paper, like receipts or invoices that an account payable might send out, or order information such as purchase orders.

Flask and SqlAlchemy

What is Flask?

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file. Flask is also extensible and doesn't force a particular directory structure or require complicated boilerplate code before getting started.

As part of this tutorial, you'll use the Bootstrap toolkit to style your application, so it is more visually appealing. Bootstrap will help you incorporate responsive web pages in your web application so that it also works well on mobile browsers without writing your own HTML,
CSS, and JavaScript code to achieve these goals. The toolkit will allow you to focus on learning how Flask works.
Flask uses the Jinja template engine to dynamically build HTML pages using familiar Python concepts such as variables, loops, lists, and so on. You'll use these templates as part of this project.
In this tutorial, you'll build a small web blog using Flask and SQLite in Python 3. Users of the application can view all the posts in your database and click on the title of a post to view its contents with the ability to add a new post to the database and edit or delete an existing post.

What is SqlAlchemy?

SQLAlchemy is a popular SQL toolkit and **Object Relational Mapper**. It is written in **Python** and gives full power and flexibility of SQL to an application developer. It is **an open source** and **cross- platform software** released under MIT license.
SQLAlchemy is famous for its object-relational mapper (ORM), using which, classes can be mapped to the database, thereby allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.
As size and performance of SQL databases start to matter, they behave less like object collections. On the other hand, as abstraction in object collections starts to matter, they behave less like tables and rows. SQLAlchemy aims to accommodate both of these principles.
For this reason, it has adopted the **data mapper pattern (like Hibernate) rather than the active record pattern used by a number of other ORMs**. Databases and SQL will be viewed in a different perspective using SQLAlchemy.
Michael Bayer is the original author of SQLAlchemy. Its initial version was released in February 2006. Latest version is numbered as 1.2.7, released as recently as in April 2018.

What is ORM?

ORM (Object Relational Mapping) is a programming technique for converting data between incompatible type systems in object-oriented programming languages. Usually, the type of system used in an Object Oriented (OO) language like Python contains non-scalar types. These cannot be expressed as primitive types such as integers and strings. Hence, the OO programmer has to convert objects in scalar data to interact with backend database. However, data types in most of the database products such as Oracle, MySQL, etc., are primary.
In an ORM system, each class maps to a table in the underlying database. Instead of writing tedious database interfacing code yourself, an ORM takes care of these issues for you while you can focus on programming the logics of the system.

Mobile Application Implementation

In this proposed project, cross-platform approach is used via Flutter. The reasons for using Flutter is it combines the advantages of the native apps and the cross-platform apps, as its performance is very high compared to the rest of the cross-platforms, and it is not far from the performance of

the Native apps, and at the same time, it surpasses the native in that it builds all kinds of applications like Android, iOS, windows and web applications which made it very popular and widely used in the last days.

Algorithm Used to Process the Speech

Automatic Speech Recognition (ASR)

We use SDK called Allan that use this algorithm. In the Alan infrastructure, all voice processing is performed in the Alan cloud. The Alan cloud is the AI-backend of the Alan Platform. This is where voice scripts are hosted and Spoken Language Understanding (SLU) and Natural Language Processing (NLP) tasks are accomplished. Under the hood, the Alan cloud engages a combination of voice AI tools and technologies to simulate human-like dialog between the user and the app. Together, they allow Alan to interpret human speech, generate responses and perform the necessary actions in the app. **The main voice technologies used by Alan are:**

- Natural language processing (NLP).
- Spoken Language Understanding (SLU).
- Automatic Speech Recognition (ASR).
- Machine Learning (ML).
- Speech-to-Text (STT) and Text-to-Speech (TTS).

How voice processing works

Alan's main goal is to match what the user says to a specific voice command in the script. To do this, Alan needs to rearrange unstructured data in the user's input so that it can be analyzed and processed at the machine level.

1. Voice commands processing starts on the client side. The Alan Client SDK captures voice stream from the user's device and sends the voice data to the Alan Cloud for processing.
2. Alan uses the Automatic Speech Recognition (ASR) engine and Speech-to-Text (STT) to get the user input and convert it to text segments.
3. With the help of Spoken Language Understanding (SLU) and Named Entity Recognition (NER) systems, Alan evaluates the phrase patterns, draws the intent and meaningful words, such as location, date and time, from the phrase. For high accuracy of matching, Alan uses the Domain Language Model for your app.
4. Alan matches the phrase to a voice command in the Alan script. This is where Alan Machine Learning (ML) algorithms are leveraged. Each phrase is given a probability score, with '1' being the most accurate match.
5. If Alan is supposed to give a response, it uses the Text-to-Speech (TTS) technology to synthesize speech that sounds natural.

ASR and Domain Language Model

During voice processing, the ASR engine converts speech to text. And one of vital components in ASR is the language model. The language model evaluates probabilities of word sequences, which allows ASR to distinguish between words that sound similar.

Beside the global language model, Alan creates a domain language model (DLM) for every app. The DLM is based on unique terms, names and phrases used in your company or domain. It lets Alan's ASR predict with high accuracy what user may say in a particular context and resolve ambiguities.
To build a DLM, you do not need to provide a large dataset with variants of utterances or to create spoken language models. Alan automatically trains on small existing datasets and generates models with phrases and intents for your app. As a result, you get an intelligent voice assistant that deeply understands the UI, workflow, and business logic of your app.
Another thing that can impact the accuracy of the ASR engine is noise. Alan uses its probabilistic NLU to face this problem. It handles speech recognition errors and ensures the voice assistant works reliably in noisy environments.

Connecting raspberry pi to mobile wirelessly

To make mobile read the data that loaded on the raspberry pi server or host we should connect the mobile to raspberry pi so can read the data. And according to our application and to make it easier for our users so we should make this connection wirelessly by connect the mobile to the raspberry pi by the mobile hotspot which he set the username and password of the hotspot mobile network on the raspberry pi and then it connects to mobile automatically.

Next chapter we will be showing the results after implementing these solutions and how accurate these implementations will be.

Chapter 5

Results

Text Recognition via OCR Output and Results

First, we test in PyCharm (Python IDE) using images for test accuracy of detection. When we use short texts as show in figure 5.1, Tesseract OCR algorithm detect Arabic and English text with 98% accuracy.

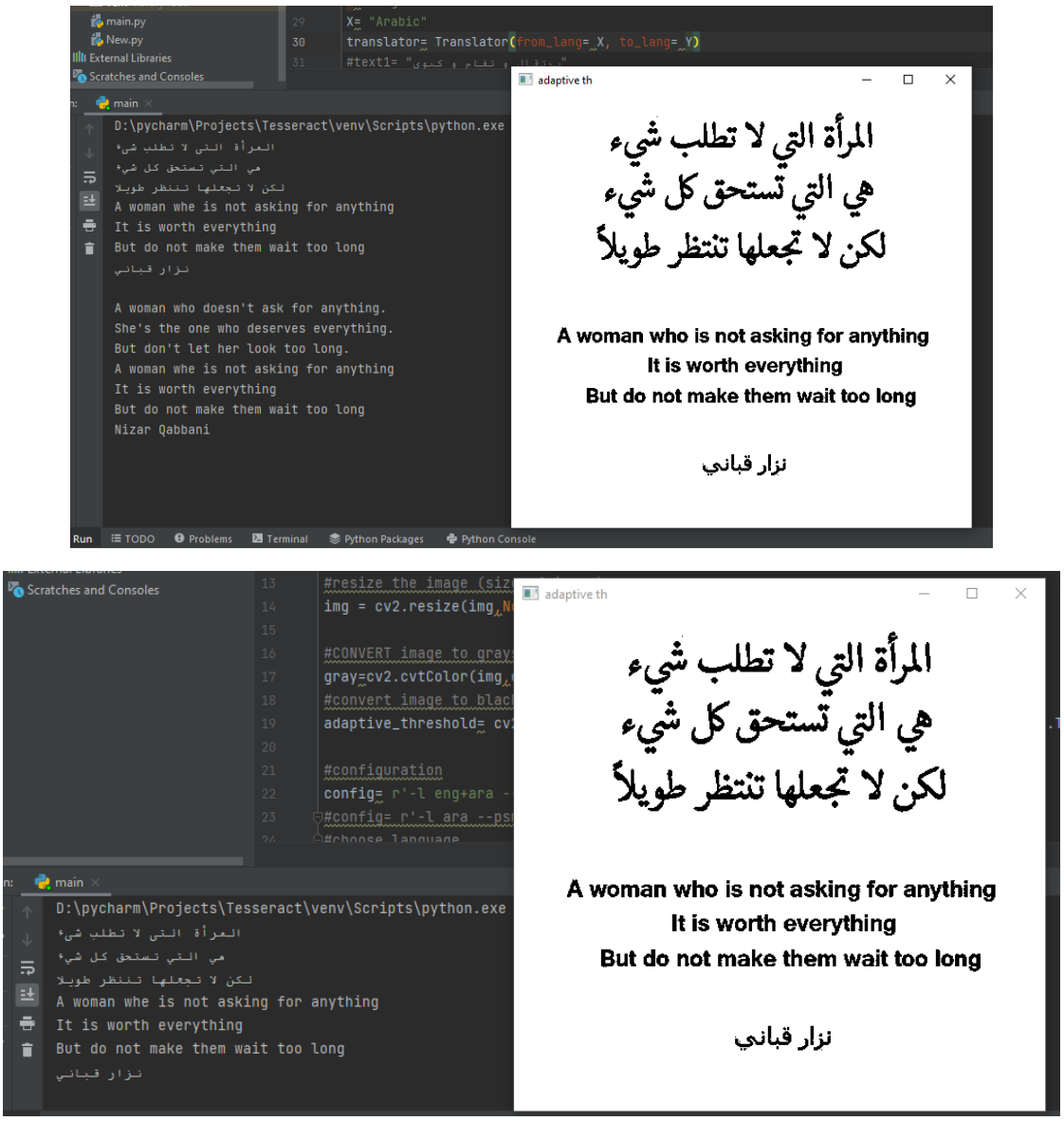
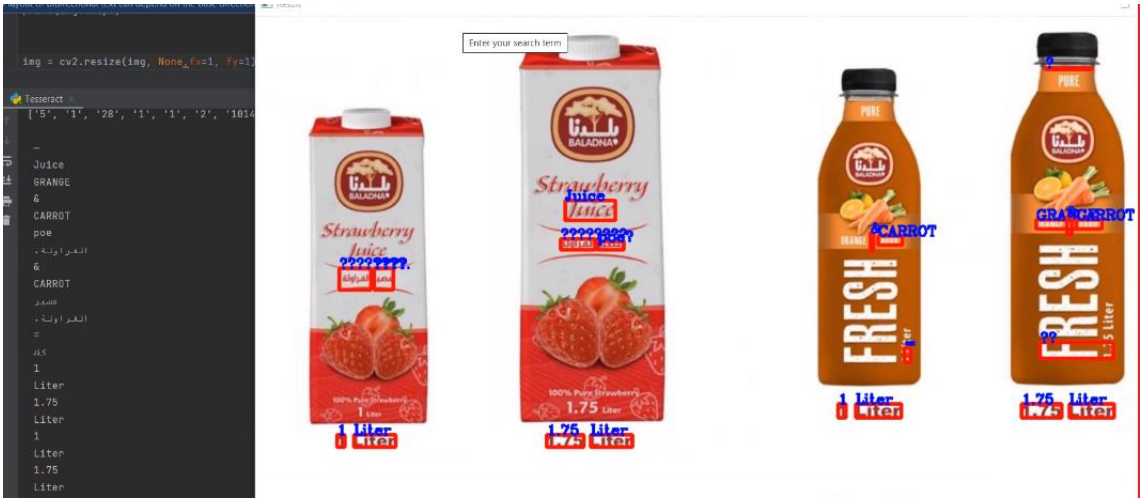
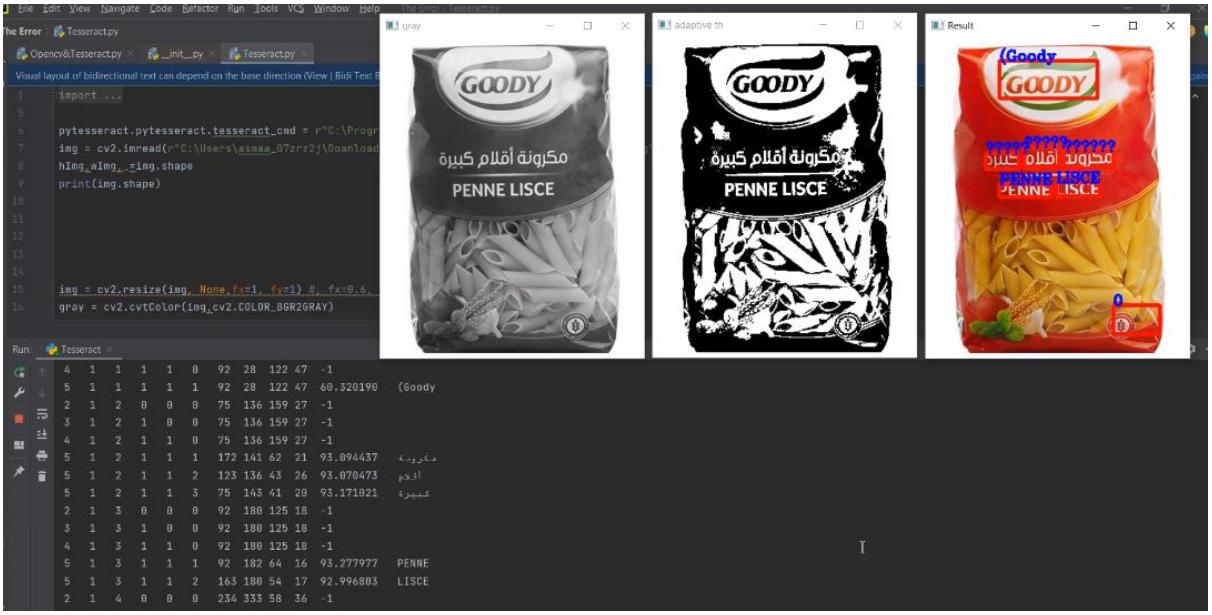


FIGURE 5-1 SHORT TEXTS

At testing supermarket products, sometimes it does not find any text to detect. We can say after many tests that accuracy of testing product can be in range 95% to 97%.





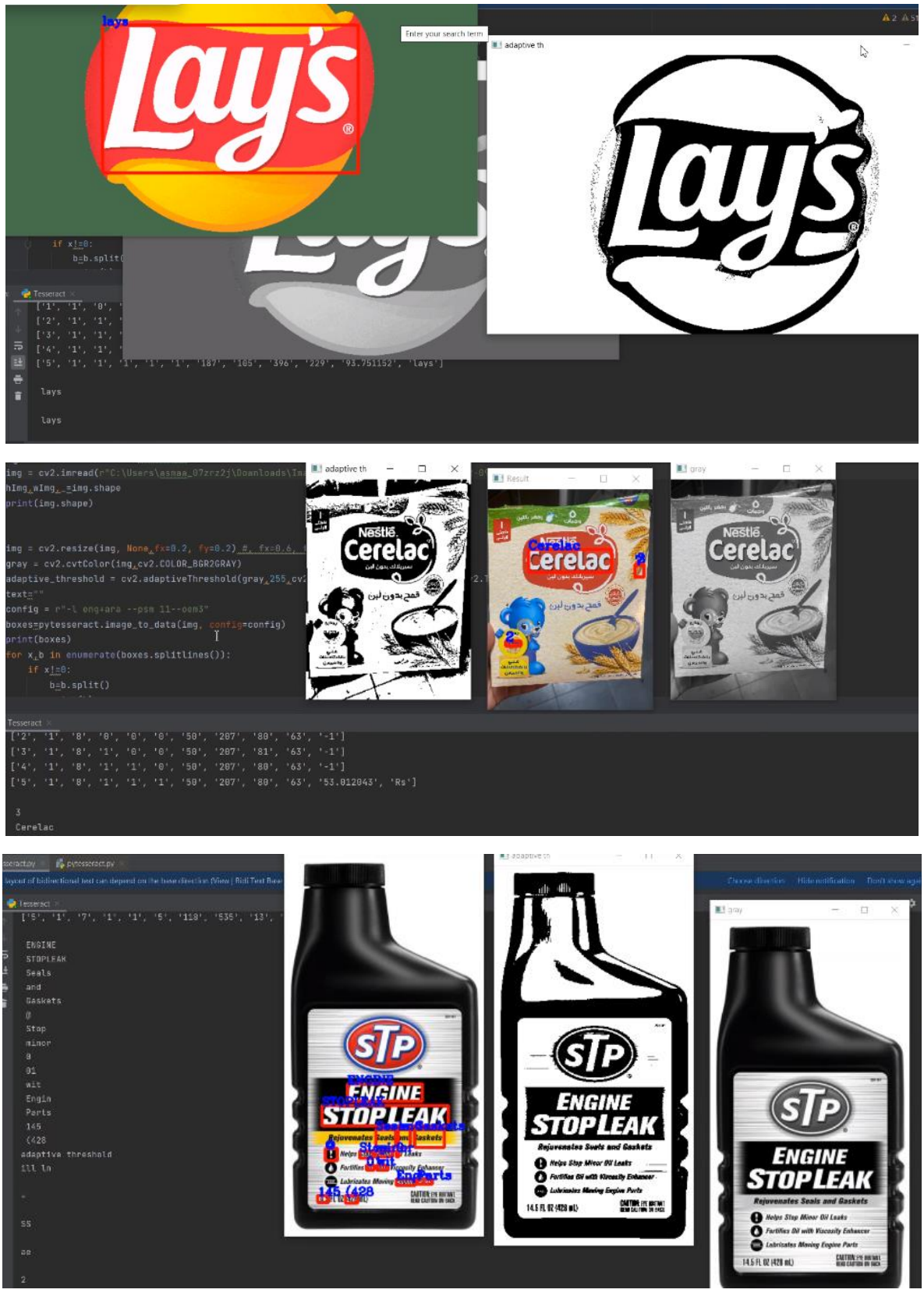
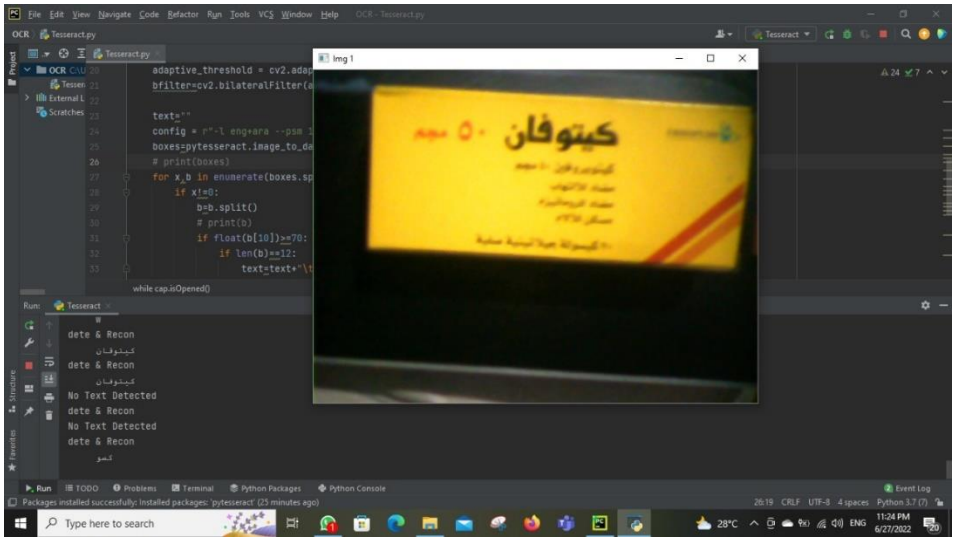


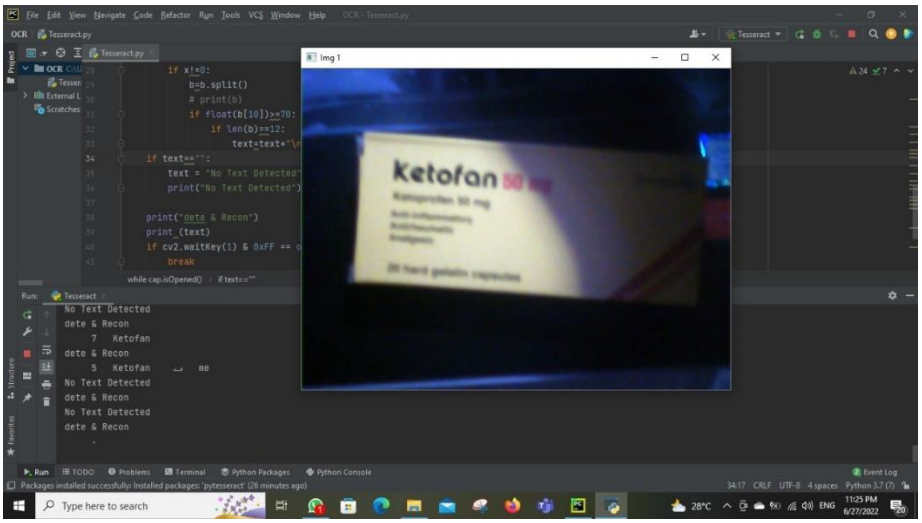
FIGURE 5-2 SUPERMARKET PRODUCTS

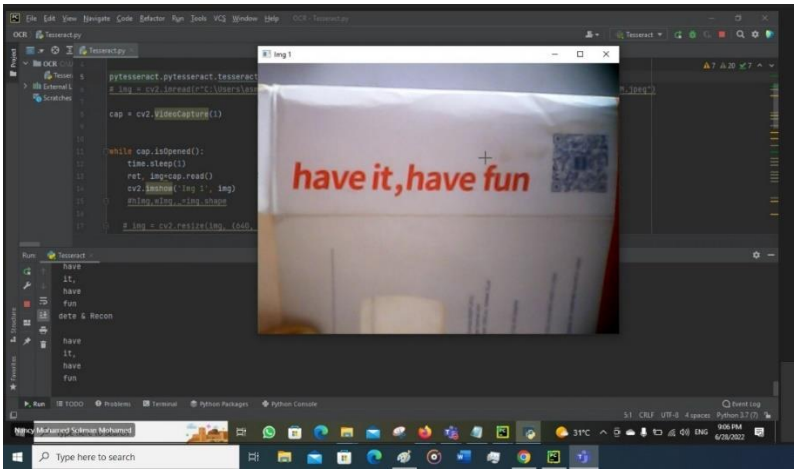
At testing labels, it detects text right most of time. We can say that the accuracy range between 97% and 98%.

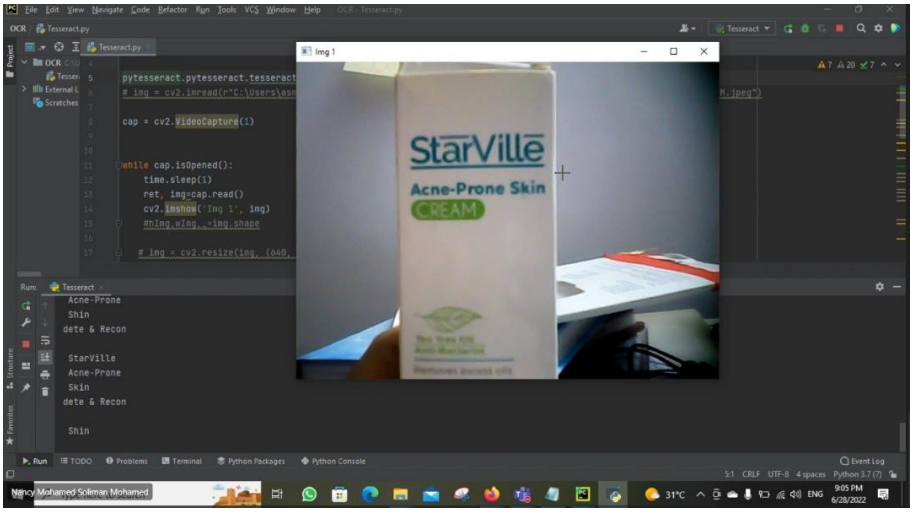
FIGURE 5-3 STOP SIGN

In those photos we are testing text recognition algorithm using video stream. The detection accuracy range between 91% and 93%.




Img 1
Ketofan
No Text Detected
dete & Recon
7 Ketofan
dete & Recon
5 Ketofan me
No Text Detected
dete & Recon
No Text Detected
dete & Recon


Img 1
have it, have fun
have
it,
have
fun
dete & Recon
have
it,
have
fun


Img 1
StarVille
Acne-Prone Skin
CREAM
Acne-Prone
Shin
dete & Recon
StarVille
Acne-Prone
Skin
dete & Recon
Shin

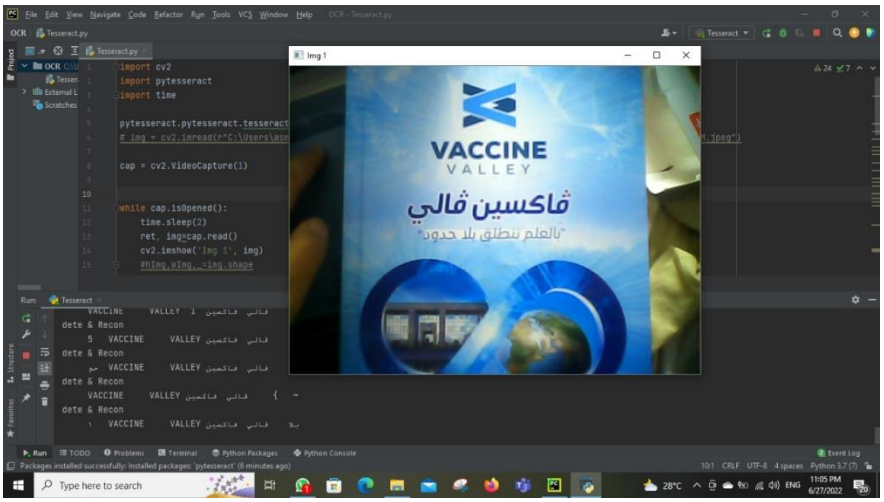


FIGURE 5-4 SHOWS SOME OTHER PRODUCTS LABELS

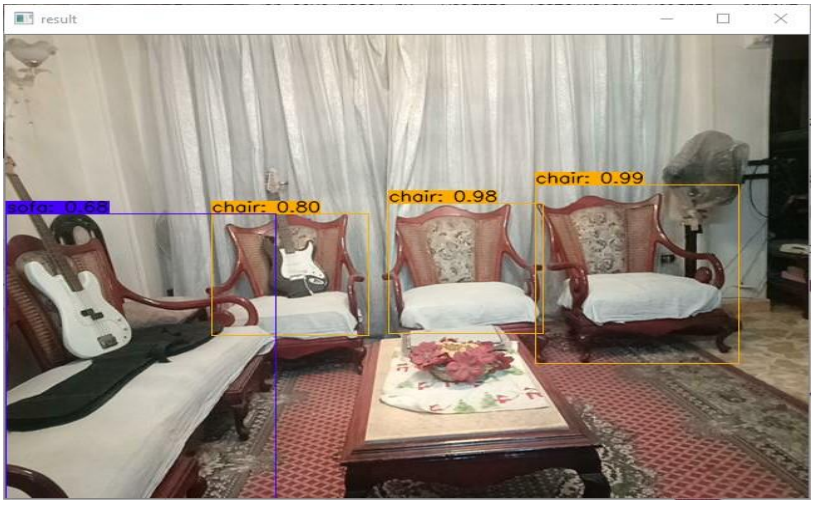
Object Detection Output and Results via YOLOv4

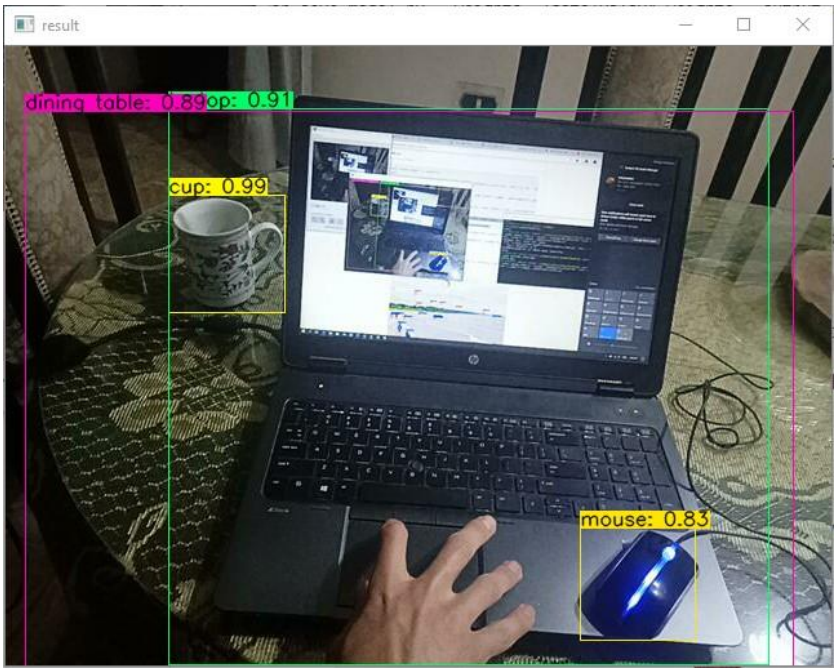
Here we are testing YOLO – the object recognition algorithm which can recognize various objects such as chairs, dining tables, laptops, etc. with accuracy of 93.3%.

The algorithm's dataset is based on COCO dataset. The COCO dataset classes for object detection and tracking include the following pre-trained 80 objects which are:

```
'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck',
'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench',
'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra',
'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis',
'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard',
'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife',
'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot
dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining
table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase',
'scissors', 'teddy bear', 'hair drier', 'toothbrush'
```

You will find below some of experimental results on COCO dataset:

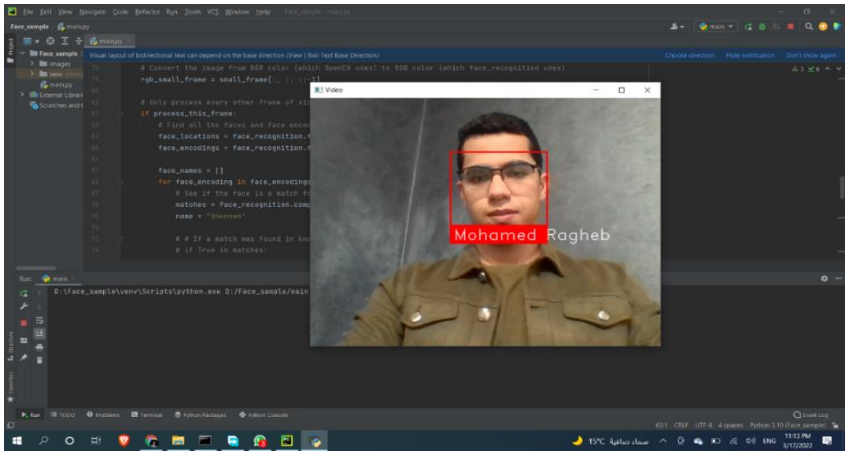





```
MINGW64:/c/Repos/yolov4-custom-functions
Object found: laptop, Confidence: 0.82, BBox Coords (xmin, ymin, xmax, ymax): 97.0, 47.0, 534.0, 458.0 , Length: 411.0,
Object found: dining table, Confidence: 0.64, BBox Coords (xmin, ymin, xmax, ymax): 64.0, 58.0, 573.0, 470.0 , Length:
09.0
FPS: 1.28
Object found: cup, Confidence: 0.99, BBox Coords (xmin, ymin, xmax, ymax): 121.0, 145.0, 196.0, 228.0 , Length: 83.0, W
Object found: chair, Confidence: 0.88, BBox Coords (xmin, ymin, xmax, ymax): 397.0, 0.0, 640.0, 140.0 , Length: 140.0,
Object found: laptop, Confidence: 0.85, BBox Coords (xmin, ymin, xmax, ymax): 46.0, 47.0, 595.0, 459.0 , Length: 412.0,
Object found: dining table, Confidence: 0.51, BBox Coords (xmin, ymin, xmax, ymax): 25.0, 64.0, 607.0, 468.0 , Length:
82.0
FPS: 1.32
Object found: cup, Confidence: 0.99, BBox Coords (xmin, ymin, xmax, ymax): 115.0, 148.0, 191.0, 232.0 , Length: 84.0, W
Object found: chair, Confidence: 0.90, BBox Coords (xmin, ymin, xmax, ymax): 386.0, 0.0, 638.0, 141.0 , Length: 141.0,
Object found: laptop, Confidence: 0.78, BBox Coords (xmin, ymin, xmax, ymax): 55.0, 43.0, 597.0, 464.0 , Length: 421.0,
Object found: dining table, Confidence: 0.56, BBox Coords (xmin, ymin, xmax, ymax): 22.0, 60.0, 615.0, 473.0 , Length:
93.0
FPS: 1.36
Object found: cup, Confidence: 0.98, BBox Coords (xmin, ymin, xmax, ymax): 117.0, 148.0, 191.0, 232.0 , Length: 84.0, W
Object found: chair, Confidence: 0.95, BBox Coords (xmin, ymin, xmax, ymax): 387.0, 0.0, 640.0, 140.0 , Length: 140.0,
Object found: laptop, Confidence: 0.83, BBox Coords (xmin, ymin, xmax, ymax): 63.0, 46.0, 595.0, 460.0 , Length: 414.0,
Object found: dining table, Confidence: 0.62, BBox Coords (xmin, ymin, xmax, ymax): 38.0, 65.0, 613.0, 466.0 , Length:
75.0
FPS: 1.33
(yolov4-gpu)
Mario@DESKTOP-D6NVANH MINGW64 /c/Repos/yolov4-custom-functions (master)
$
```

FIGURE 5-5 EXPERIMENTING THE OBJECT RECOGNITION ALGORITHM

Face Detection Output and Results



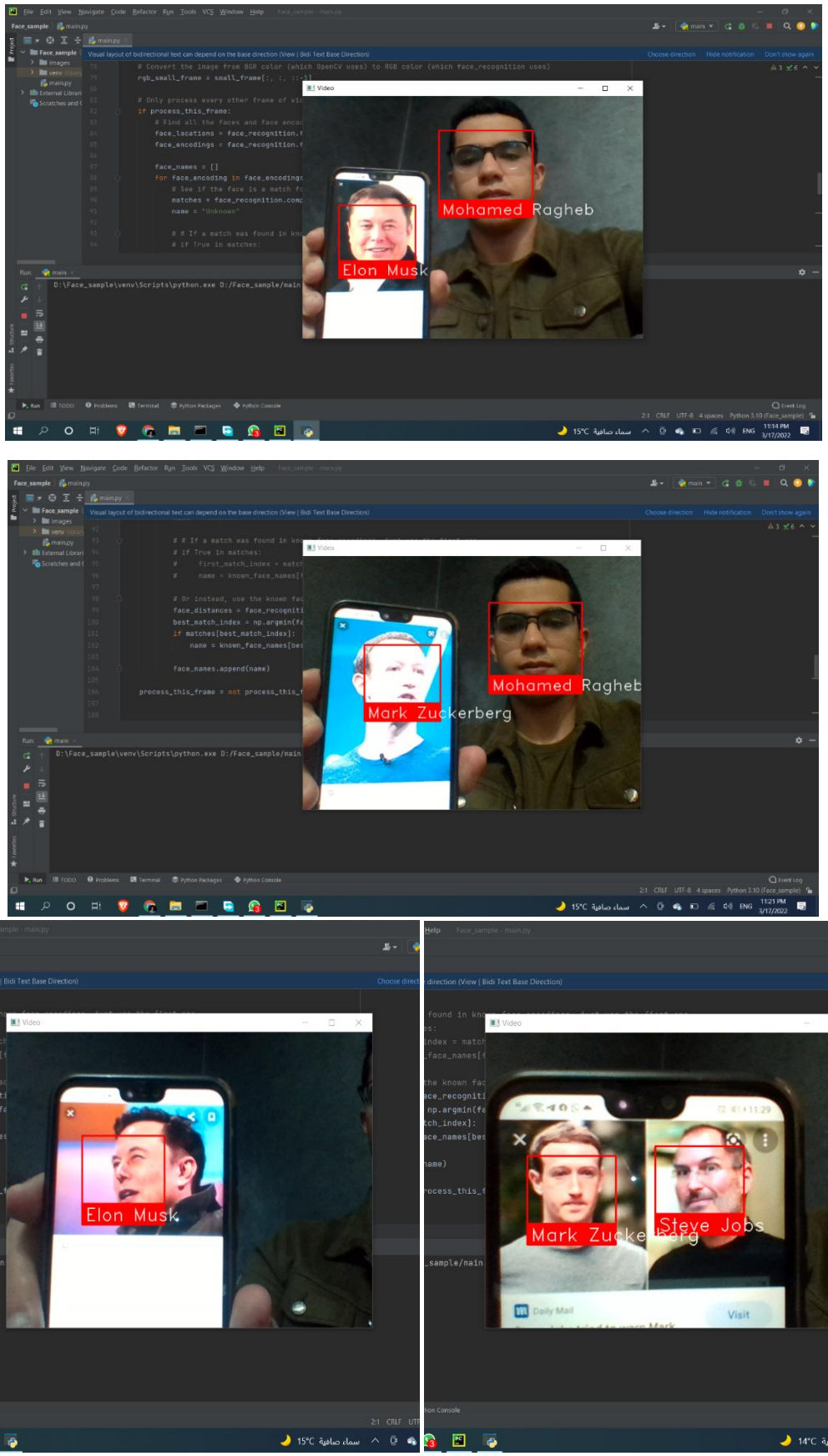


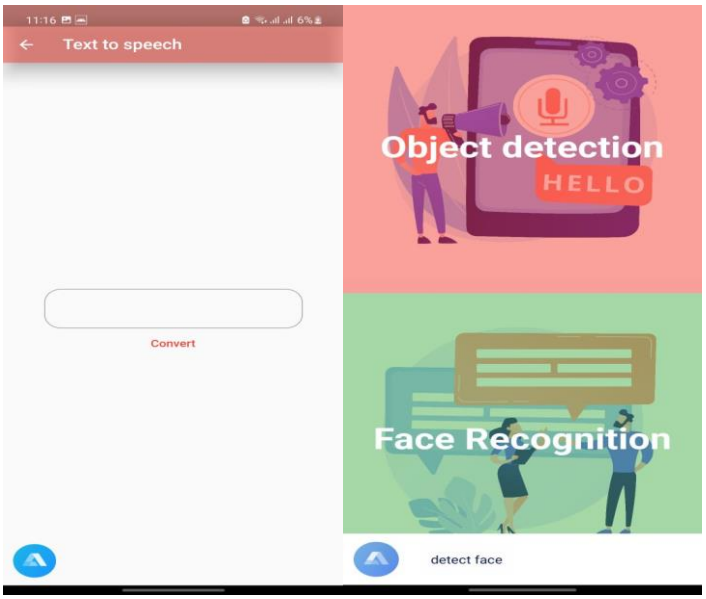
FIGURE 5-6 EXPERIMENTING THE FACE DETECTION ALGORITHM

Distance Estimation Results

we test measuring distance using to camera from video steam. After many times of testing the accuracy of Distance Estimation in range 91% to 93%. When any object near to the blind people in specific range it will send warning message to blind people through mobile.

FIGURE 5-7 EXPERIMENTING DISTANCE MEASUREMENT ALGORITHM

Mobile Application Output and Results



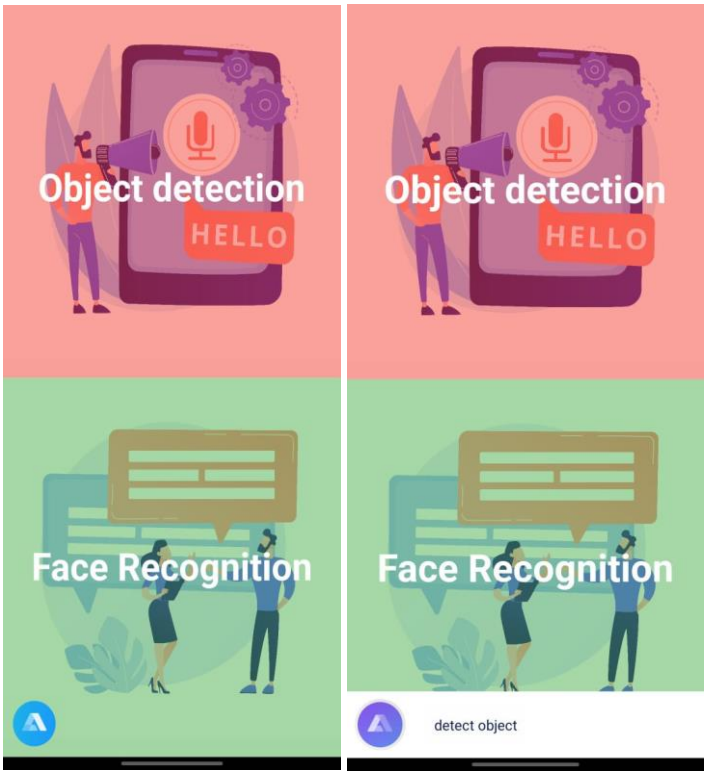


FIGURE 5-8 SCREENSHOTS OF THE MOBILE INTERFACE

Next chapter is the last chapter, and it will be the conclusion of this proposed project and the future work as well as the references.

Chapter 6

Conclusion & Future Work

CONCLUSION

Considering all the above, it's not a surprise that living with a visual impairment might signify, often, living in isolation. Dealing with sight loss, already, is a challenge in itself. The lack of emotional support at diagnosis centers, the limited accessibility to activities and information, the societal stigma, and the lack of unemployment, are all factors frequently leading blind or low vision individuals in isolation. This last point illustrates how the problem for the visually impaired is not their blindness or lower vision but their segregation from anyone else.

So, the main goal of the project is to help the blind to lead a healthy and active life so he can be treated as a normal person maybe find a job and get his dependency.

That was achieved by our project:

Our main goal in the project is to maintain high performance with the less hardware as much as possible.

In addition, we offer a state-of-the-art detector which is faster (FPS) and more accurate (MS COCO AP50...95 and AP50) than all available alternative detectors. The original concept of one-stage anchor-based detectors has proven its viability. We have verified many features and selected for use such of them for improving the accuracy of both the classifier and the detector. These features can be used as best-practice for future studies and development.

FUTURE WORK

In Text recognition mode, book mode will be added and detecting handwritten text to improve the accuracy of recognition.

In Distance Estimation mode, we will be using camera that are made specifically for distance measuring for more accuracy and locate the objects around the visually impaired person.

In Mobile, Authentication will be added to deal with multiple users, translation to Arabic and other languages.

References

1. Ait Hamou Aadi, F., & Sadiq, A. (2020). Proposed real-time obstacle detection system for visually impaired assistance based on deep learning. *International Journal of Advanced Trends in Computer Science and Engineering, 9*. doi:10.30534/ijatcse/2020/357942020

2. Al-amri, S. S., Kalyankar, N., & S.D., K. (2010, May). Image Segmentation by Using Thershod Techniques. *JOURNAL OF COMPUTING, 2*(5). Retrieved from https://arxiv.org/ftp/arxiv/papers/1005/1005.4020.pdf

3. *All Tesseract OCR options*. (2020, July 28). Retrieved from Muthukrishnan: https://muthu.co/all-tesseract-ocr-options/

4. Anderson, M. (2021, June 30). *OCR algorithms: a complete guide*. Retrieved from itransition: https://www.itransition.com/blog/ocr-algorithm

5. awed, K., Morris, J., Khan, T., & Gimel'farb, G. (2009). Real Time Rectification for Stereo Correspondence. *2*, 277-284. doi:10.1109/CSE.2009.473

6. Azar, Y. M. (2016, September). *FULLTEXT01*.

7. Bah, S. M., & Ming, F. (2020). An improved face recognition algorithm and its application in attendance management system. *Array, 5*. Retrieved from https://reader.elsevier.com/reader/sd/pii/S2590005619300141?token=EAA08E48D97B851B7ABB7E92019AB9142563F4806ED888C628FE787E3A40F9D56A8DEEA307D16D73D40579EE56A217CD&originRegion=eu-west-1&originCreation=20220713083222

8. Baltrušaitis, T., Robinson, P., & Morency, L.-P. (2016). OpenFace: An open source facial behavior analysis toolkit. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1-10). IEEE. doi:10.1109/WACV.2016.7477553

9. *Basic Drawing*. (n.d.). Retrieved from OpenCV: https://docs.opencv.org/4.x/d3/d96/tutorial_basic_geometric_drawing.html

10. Basulto, D. (2014, March 25). *Artificial intelligence is the next big tech trend. Here's why.* Retrieved from The Washington Post: https://www.washingtonpost.com/news/innovations/wp/2014/03/25/artificial-intelligence-is-the-next-big-tech-trend-heres-why/

11. Bhandari, A. (2020, May 16). *Build your own Optical Character Recognition (OCR) System using Google's Tesseract and OpenCV*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/05/build-your-own-ocr-google-tesseract-opencv/

12. Blaettler, K. G. (2020, December 28). *How to Calculate Focal Length of a Lens*. Retrieved from SCIENCING: https://sciencing.com/calculate-focal-length-lens-7650552.html

13. *Camera Calibration*. (n.d.). Retrieved from OpenCV: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

14. *Camera Calibration and 3D Reconstruction*. (n.d.). Retrieved from OpenCV: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

15. Chen, H., & Zhang, B. (2019). Application of Automatic Speech Recognition (ASR) Algorithm in Smart Home. *Journal of Physics: Conference Series*. doi:10.1088/1742-6596/1237/2/022133

16. CHEN, X., JIN, L., ZHU, Y., LUO, C., & WANG, T. (2020, December 3). Text Recognition in the Wild: A Survey. *1*. Retrieved from https://arxiv.org/pdf/2005.03492.pdf

17. Chena, X., Wang, T., Zhu, Y., Jin, L., & Luo, C. (2019). Adaptive embedding gate for attention-based scene text recognition. *Neurocomputing*, 261-271. Retrieved from https://www.researchgate.net/publication/337754657_Adaptive_Embedding_Gate_for_Attention-Based_Scene_Text_Recognition/fulltext/5e73d5cb92851c35875983a9/Adaptive-Embedding-Gate-for-Attention-Based-Scene-Text-Recognition.pdf

18. Cilloni, T., Wang, W., Walter, C., & Fleming, C. (2022, January). Ulixes: Facial Recognition Privacy with Adversarial Machine Learning. *Proceedings on Privacy Enhancing Technologies, 2022*, 148-165. doi:10.2478/popets-2022-0008

19. Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., . . . Ng, A. Y. (n.d.). Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning. Retrieved from https://www.cs.utexas.edu/~dwu4/papers/TextRecog.pdf

20. Collins, R. (n.d.). *Stereo Algorithms.* Retrieved from https://www.cse.psu.edu/~rtc12/CSE486/lecture09.pdf

21. Cristianini, N., & Ricci, E. (n.d.). *Support Vector Machines.* Boston: Springer. doi:https://doi.org/10.1007/978-0-387-30162-4

22. *Depth Map from Stereo Images*. (n.d.). Retrieved from OpenCV: https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html

23. El abbadi, N., Dahir, N., Alyasseri, Z., & Alkareem, A. (2008). SKIN TEXTURE RECOGNITION USING NEURAL NETWORKS. *ACIT*.

24. *Epipolar Geometry*. (n.d.). Retrieved from OpenCV: https://docs.opencv.org/4.x/da/de9/tutorial_py_epipolar_geometry.html

25. Gerig, G. (2012). *Image Rectification (Stereo).* Retrieved from http://www.sci.utah.edu/~gerig/CS6320-S2013/Materials/CS6320-CV-F2012-Rectification.pdf

26. Gupta, A. (2021, April 30). *Top Python Libraries For Image Processing In 2021*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/04/top-python-libraries-for-image-processing-in-2021/

27. Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory.* Retrieved from http://www.bioinf.jku.at/publications/older/2604.pdf

28. *Image Thresholding*. (n.d.). Retrieved from OpenCV: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

29. Imran, M. A., Miah, M. S., Rahman, H., Bhowmik, A., & Karmaker, D. (2015). Face Recognition using Eigenfaces. *International Journal of Computer Applications, 118*, 12-16. Retrieved from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.3122&rep=rep1&type=pdf

30. Kakadiaris, I. A., Passalis, G., Toderici, G., Murtuza, M. N., Lu, Y., Karampatziakis, N., & Theoharis, T. (2007). Three-Dimensional Face Recognition in the Presence of Facial Expressions: An

Annotated Deformable Model Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 640 - 649. Retrieved from https://ieeexplore.ieee.org/document/4107568/authors#authors

31. Kazemi, V., & Sullivan, J. (2014). One Millisecond Face Alignment with an Ensemble of Regression Trees. doi:10.13140/2.1.1212.2243

32. King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, 1755-1758. Retrieved from https://www.jmlr.org/papers/volume10/king09a/king09a.pdf

33. Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., . . . Wen, S. (2020). PP-YOLO: An Effective and Efficient Implementation of Object Detector. *ArXiv*.

34. M.F.Saaid, A.M.Mohammad, & Ali, M. M. (2016). Smart cane with range notification for blind people. *IEEE International Conference on Automatic Control and Intelligent Systems*, 225-229.

35. M.N., G., H.V., S., S., S., J., A. S., & H.C., C. (2019). Survey on Smart Reader for Blind and Visually Impaired (BVI). *Indian Journal of Science and Technology, 12*(48), 1-4.

36. Madhuram, M., & Parameswaran, A. (2018, September). A Text Extraction Approach towards Document Image Organisation forthe Android Platform. *International Research Journal of Engineering and Technology, 5*(9), 1560-1565.

37. Masood, A. (n.d.). Stereo Pi: Portable Digital Stereo Camera. Retrieved from http://stanford.edu/class/ee367/Winter2016/Masood_Report.pdf

38. Mind, P., Palkar, G., Mahamuni, A., & Sahare, S. (2021, June). Smart Stick for Visually Impaired. *International Journal of Engineering Research & Technology, 10*(6), 196-198.

39. Mishra, R., Sharma, A., & Ramaiah, N. S. (2020, May 25). Third Eye for Blind. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3613781

40. Moncef, A., & M'hamed, A. K. (2021). Localization and Navigation System for Blind Persons Using Stereo Vision and a GIS. 365-376. doi:10.1007/978-981-33-6893-4_35

41. Mondal, M. A., & Ali, M. (2017). Performance Review of the Stereo Matching Algorithms. *American Journal of Computer Science and Information Engineering, 4*, 7-15.

42. MVA 2011 : IAPR Conference on Machine Vision Applications. (2011). *The Twelfth IAPR Conference on Machine Vision Applications .* Japan.

43. Nanayakkara, S., Shilkrot, R., Yeo, K. P., & Maes, P. (2013). *EyeRing: A Finger-Worn Input Device for SeamlessInteractions with our Surroundings.* Germany.

44. Natta, M. V., Chen, P., Herbek, S., Jain, R., Kastelic, N., Katz, E., . . . Vattikonda, N. (2020). The rise and regulation of thermal facial recognition technology during the COVID-19 pandemic. *Journal of Law and the Biosciences*.

45. Nayak, M., & Nayak, A. K. (2014). Odia Characters Recognition by Training Tesseract OCR Engine. *International Journal of Computer Applications*, 25-30.

46. Nigin, J. (2018). Research on Smart Speech Module Based Smart Home System [J]. *Electronics Production*, 25-27.

47. *OPENCV AND DEPTH MAP ON STEREOPI TUTORIAL*. (n.d.). Retrieved from StereoPi: https://stereopi.com/blog/opencv-and-depth-map-stereopi-tutorial

48. Pal, S. (2019, March 25). *16 OpenCV Functions to Start your Computer Vision journey (with Python code)*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/?utm_source=blog&utm_medium=build-your-own-ocr-google-tesseract-opencv

49. Patel, K., Shah, H., & Kher, R. (2015, December). Face Recognition Using 2DPCA and ANFIS Classifier. *Advances in Intelligent Systems and Computing, 336*, 1-12. doi:10.1007/978-81-322-2220-0_1

50. Pengyuan, Z., Zhe, J., Wei, H., Xin, J., & Weisheng, H. (2017). Design and optimization of a low resource speech recognition system. *Journal of Tsinghua University(Science and Technology*, 147. doi:10.16511/j.cnki.qhdxxb.2017.22.006

51. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). ou Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779-788). IEEE. doi:10.1109/CVPR.2016.91

52. Rosebrock, A. (2014, November 17). *Non-Maximum Suppression for Object Detection in Python*. Retrieved from Pyimagesearch: https://pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/

53. Rosebrock, A. (2017, November 6). *Deep learning: How OpenCV's blobFromImage works* . Retrieved from Pyimagesearch: https://pyimagesearch.com/2017/11/06/deep-learning-opencvs-blobfromimage-works/

54. Rosebrock, A. (2018, September 17). *OpenCV OCR and text recognition with Tesseract*. Retrieved from pyimagesearch: https://pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract/

55. S., A. (2016). *Overview of Tesseract OCR engine.* National Institute of Technology, Calicut.

56. S., U. B., & K., S. (2019, December). A Novel Approach Towards Implementation Of Optical Character Recognition Using LSTM And Adaptive Classifier. *Journal of Engineering & Management, 3*(2).

57. Sadekar, K., & Mallick, S. (2020, February 25). *Camera Calibration using OpenCV*. Retrieved from LearnOpenCV: https://learnopencv.com/camera-calibration-using-opencv/

58. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 815-823). IEEE. doi:10.1109/CVPR.2015.7298682

59. Selvi.R, S., D.Sivakumar, Sandhya.J.S., Sowmiya.S, S., Ramya.S, & Raja.S, K. S. (2019). Face Recognition Using Haar - Cascade Classifier for Criminal Identification. *International Journal of Recent Technology and Engineering, 7*, 1871-1876.

60. Sharif, M., Javed, M. Y., & Mohsin, S. (2012). Face Recognition Based on Facial Features. *Research Journal of Applied Sciences, Engineering and Technology*, 2879-2886.

61. Sharma, P., L., S. S., & S.Chatterji. (2015, January). A REVIEW ON OBSTACLE DETECTION AND VISION. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY*.

62. Shuailin, Z. (2017). An Improved Key Word Recognition Algorithm on Smart Home Speech [J]. *Electronics Technology*, 5-8.

63. Singh, R., Kamat, R. K., & Chisti, F. (2021, June). Third Eye for Blind Person. *8*(1).

64. Singh, R., Yadav, C. S., Verma, P., & Yadav, V. (2010, June). Optical Character Recognition (OCR) for Printed Devnagari Script Using Artificial Neural Network. *International Journal of Computer Science & Communication, 1*, 91-95.

65. Smith, R. (n.d.). *An Overview of the Tesseract OCR Engine.* Retrieved from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf

66. Sun, X., Jiang, Y., Y. J., Fu, W., & Yan, S. (2018). Distance Measurement System Based on Binocular Stereo Vision. *Earth and Environmental Science.* IOP Publishing. Retrieved from https://iopscience.iop.org/article/10.1088/1755-1315/252/5/052051/pdf

67. Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1701-1708). IEEE. doi:10.1109/CVPR.2014.220

68. *Understanding LSTM Networks*. (2015, August 27). Retrieved from colah's blog: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

69. *Universal eye health: a global action plan 2014-2019.* (2013). Spain: World Health Organization.

70. Viola, P., & Jones, M. (2001). *Robust Real-time Object Detection.* Canada: SECOND INTERNATIONAL WORKSHOP ON STATISTICAL AND COMPUTATIONAL THEORIES OF VISION. Retrieved from https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-IJCV-01.pdf

71. *What is computer vision?* (n.d.). Retrieved from IBM: https://www.ibm.com/topics/computer-vision

72. (n.d.). *What is OCR?* NECC. Retrieved from https://www.necc.mass.edu/wp-content/uploads/accessible-media-necc/uncategorized/resources/What-is-OCR.pdf

73. *WHAT IS OPENCV?* (2021, September 2). Retrieved from YoungWonks: https://www.youngwonks.com/blog/What-is-OpenCV

74. Yefen, Y., & Chengjing, Y. (2017). A GSM Based Smart Home Speech Control System [J]. *Computer System Application*, 68-72.

75. Yunanto, A., & Murti, D. H. (2016). Face recognition based on Extended Symmetric Local Graph Structure. *2016 International Conference on Information & Communication Technology and Systems (ICTS)*, (pp. 80-84). doi:10.1109/ICTS.2016.7910277

76. Zaarane, A., Slimani, I., Okaishi, W., Issam, A., & Hamdoun, A. (2020). Distance measurement system for autonomous vehicles using stereo camera. *Array, 5*. doi:10.1016/j.array.2020.100016

77. Zaborowska, Ł. (n.d.). *Focal Length Calculator*. Retrieved from omniCALCULATOR: https://www.omnicalculator.com/other/focal-length

78. Zamir, M. F., Khan, K. B., Khan, S. A., & Rehman, E. (2020). Smart Reader for Visually Impaired PeopleBased on Optical Character Recognition. 79-89. doi:10.1007/978-981-15-5232-8_8

79. Zelic, F., & Sable, A. (2022, June). *How to OCR with Tesseract, OpenCV and Python*. Retrieved from Nanonets : https://nanonets.com/blog/ocr-with-tesseract/

80. Zhao, X., & Wei, C. (2017). A real-time face recognition system based on the improved LBPH algorithm. In *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)* (pp. 72-76). Singapore: IEEE. doi:10.1109/SIPROCESS.2017.8124508

81. Zhou, R. (2019, August). Obstacle Avoidance Gloves for the Blind Based on Ultrasonic Sensor. *Intelligent Control and Automation, 10*(3). doi:10.4236/ica.2019.103007

82. Zivingy, M. (2013). Object distance measurement by stereo vision. *nternational Journal of Science and Applied Information Technology (IJSAIT), 2*, 5-8.