A MINI PROJECT REPORT ON

Object Detection in Autonomous Cars

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

By

S.V. Mahati (19A81A0554)

M.R. Vijay Narasimha (19A81A0533)

G.S. Pujitha (19A81A0518)

J.S. Vishnu Vardhan (19A81A0522)

Under the Esteemed Supervision of

Dr. D. Jaya Kumari, M.Tech.,Ph.D.

Professor & HOD.

Department of Computer Science and Engineering (Accredited by N.B.A.)
SRI VASAVI ENGINEERING COLLEGE(Autonomous)
(Affiliated to JNTUK, Kakinada)
Pedatadepalli, Tadepalligudem-534101, A.P 2020-21

SRI VASAVI ENGINEERING COLLEGE (Autonomous)

Department Of Computer Science and Engineering

Pedatadepalli, Tadepalligudem

Certificate

This is to certify that the Project Report entitled "**Object Detection in Autonomous Cars**" submitted by **S.V. MAHATI (19A81A0554), M.R.VIJAY NARASIMHA (19A81A0533), G.S. PUJITHA (19A81A0518), J.S. VISHNU VARDHAN (19A81A0522)** for the award of the degree of Bachelor of Technology in the Department of Computer Science and Engineering during the academic year 2021-2022.

Name of Project Guide

Dr. D Jaya Kumari M.Tech.,Ph.D..
Professor & HOD.

Head of the Department

Dr. D Jaya Kumari M.Tech.,Ph.D..
Professor & HOD.

External Examiner

DECLARATION

We hereby declare that the project report entitled "**Object Detection in Autonomous Cars**" submitted by us to Sri Vasavi Engineering College(Autonomous), Tadepalligudem, affiliated to JNTUK Kakinada in partial fulfilment of the requirement for the award of the degree of B.Tech in Computer Science and Engineering is a record of Bonafide project work carried out by us under the guidance of **Dr. D Jaya Kumari,Professor & HOD**. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or University.

Project Associates

S.V. Mahati (19A81A0554)

M.R. Vijay Narasimha (19A81A0533)

G.S. Pujitha (19A81A0518)

J.S. Vishnu Vardhan (19A81A0522)

ACKNOWLEDGEMENT

First and foremost, we sincerely salute to our esteemed institute **SRI VASAVI ENGINEERING COLLEGE,** for giving us this golden opportunity to fulfill our warm dream to become an engineer.

Our sincere gratitude to our project guide **Dr. D Jaya Kumari,Professor & HOD**, Department of Computer Science and Engineering**,** for her timely cooperation and valuable suggestions while carrying out this project.

We express our sincere thanks and heartful gratitude to **Dr. D. Jaya Kumari**, Professor & Head of the Department of Computer Science and Engineering, for permitting us to do our project.

We express our sincere thanks and heartful gratitude to **Dr. G.V.N.S.R. Ratnakara Rao**, Principal, for providing a favourable environment and supporting us during the development of this project.

Our special thanks to the management and all the teaching and non-teaching staff members, Department of Computer Science and Engineering, for their support and cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

We would like to express our gratitude to our parents, friends who helped to complete this project.

Project Associates

S.V. Mahati (19A81A0554)

M.R. Vijay Narasimha (19A81A0533)

G.S. Pujitha (19A81A0518)

J.S. Vishnu Vardhan (19A81A0522)

ABSTRACT

Autonomous vehicles have been considered as one of the most important trending topics in the domain of intelligent transportation systems. It is expected that most of the leading companies will launch their fully self-driving vehicles by the end of the next decade. Such a technology must assure a high safety level on road networks to reduce the number of accidents caused by human errors. All the built-in technologies must be integrated and complemented to achieve these goals. Automated object and obstacle detection is one of the main research tasks that must be taken.

Algorithm

• SSD- MobileNetv2 , SSD- MobileNetv3

• YOLO v1, v2

Dataset COCO 80 classes

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

According to the recent survey report from the NHTSA i.e, National Highway Traffic Safety Administration, it is confirmed that 94% of the road accidents are caused only due to the human errors. In some situation the driver cannot be able to identify the objects from his position properly due to the following reasons like lack of attention while driving, due to fog weather, reflection of lights and anonymous objects etc. In these situations, it leads to the fatal accidents. The vehicle driving requires a more driving skills and experience from the driver. To overcome above problems the Autonomous Vehicles(AV) came into existence, which uses Advance Driving Assistance System (ADAS). This system grasp surrounding environment and provides solution for the enhancement of the road safety, reducing traffic, emissions and for the passenger comfort. Autonomous vehicles are made up of the different sensors, which are used to persist the surrounding environment while driving. These vehicles mainly reduce the human work during driving and improves the driving safety compared to vehicles driven by humans.

1.2 MOTIVATION

Self-driving systems are commonly categorized into three subsystems: perception, planning, and control. The perception system is responsible for translating raw sensor data into a model of the surrounding environment, the planning system makes purposeful decisions based on the environment model, and the control system executes planned actions. The objective of this thesis is to study the perception problem in the context of real-time object detection for autonomous vehicles. Although we are standing at the edge to a new era of automated driving, this technology is still far from being mature. Self-driving cars have to interact and operate in very unpredictable dynamic environments with multiple actors such as pedestrians, other vehicles, and animals. Therefore, there is a need to provide autonomous vehicles with reliable perception systems to facilitate and improve the vehicle's ability to understand the surrounding environment. In the past, self-driving cars mostly have relied on technologies like Radar, LiDAR, GPS and other sensors when mapping the surroundings. However, compared to these sensors cameras are substantially less expensive, and therefore it would be, from a purely financial point of view, highly beneficial to incorporate them more in self-driving systems. Moreover, cameras have a higher resolution than any other used sensors, and there exist more available image-based datasets.

With the rise of deep learning, data has proven to be both the limiting and driving factor when training DNNs, particularly within CV. Creating diverse real-life datasets for driving scenarios is both time-consuming and expensive. Instead, virtual datasets can be automatically constructed and labeled with much higher speed and accuracy, and thereby they could help to solve the big deficit in labeled data for self-driving systems.Therefore, if real-time object detectors trained on virtual images is shown to have comparable performance to realtime object detectors trained on real-life collected data, or if synthetic data can be used as complementary data during training, it could accelerate the development of self-driving systems

1.3 SCOPE

The scope of this project is to study and analyze the problems faced in the Perception subsystem in the domain of detecting objects for autonomous cars. Previously, technologies like Radar, LiDAR, GPS and various other sensors had been employed for Driverless cars for mapping the surroundings of the car. However, in the recent past, some deep neural network (DNN) architectures like YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector) have been developed which are capable of detecting objects even when live video is considered as the input, thus having potential to be included as a part of the Driverless car systems

2. LITERATURE SURVEY

1) Object Detection for Autonomous Vehicles Gene Lewis Stanford University Stanford, CA

In this work, they have examined an approach to deep object detection that makes bounding box predictions for an image without the need for expensive preprocessing or expensive deep evaluations; the resulting DIY network, SimpleNet, gives reasonable prediction accuracy and runs in near-real-time. Though by no means state-of-the-art, SimpleNet is an interesting look into the power of loss functions; while most networks derive power from depth of layers, we look to derive power from suitable loss functions with a limited number of parameters and functional representation. Interesting future work includes trying to extend the training loss function to more accurately capture the desired metrics at hand, increasing the prediction speed by sharing computation, and increasing the size of data trained and tested on.

2) Chengji Liu, Yufan Tao, Jiawei Liang, Kai Li1, Yihang Chen Object Detection Based on YOLO Network

The experiment showed that the model trained with the standard sets does not have good generalization ability for the degraded images and has poor robustness. Then the model was trained using degraded images which resulted in improved average precision. It was proved that the average precision for degraded images was better in general degenerative model compared to the standard model.

3) Rumin Zhang, Yifeng Yang An Algorithm for Obstacle Detection based on YOLO and Light Filed Camera

The images of the common obstacles were labeled and used for training YOLO. The object filter is applied to remove the unconcern obstacle. Different types of scene, including pedestrian, chairs, books and so on, are demonstrated to prove the effectiveness of this obstacle detection algorithm.

CHAPTER 3

SYSTEM STUDY AND ANALYSIS

3. SYSTEM STUDY AND ANALYSIS

3.1 Problem Statement

Human error is one of the main threats in cause of major road accidents, so we would like to develop the object detection system using YOLO algorithm applied on image data and video data to detect objects,, which will help in asafe driving.

3.2 Existing System

- In existing system only prediction based on input image is possible.

3.3 Proposed System

In our proposed system, we are trying to build detection through camera and video. This is an extension going to be implemented in our project.

3.4 Advantages of Proposed System

1. By using proposed application, it takes less time delay in detecting upcoming obstacles.

2. It is very less cost to deploy the application using software programming.

3. This will generate always accurate results when trained on a large dataset.

3.5 Functional Requirements

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Generally, functional requirements are expressed in the form “system shall do ”. The plan for implementing functional requirements is detailed in the system design.

In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements.

The hierarchy of functional requirements is: user/stakeholder request -> feature -> use case -> business rule. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. Functional requirements may be technical details, data manipulation and other specific functionality of the project is to provide the information to the user. The following are the Functional requirements of our system:

1. The system should get an input image and predict the object with confidence value.

2. A novel framework to exploit the user's social activities for personalized image search, such as annotations and the participation of interest groups.

3.6 Non-Functional Requirements

In our project we are using YOLO algorithm for object detection. So, accordingly the non-functional requirements are

1. The system divide the image into S X S grid. If center of an object falls into a grid cell, that cell is responsible for detecting the object.
2. Each grid cell predicts B bounding boxes and confidence scores in the range 50-80 %for those boxes.

Availability: A system's "availability" or "uptime" is the amount of time that is operational and available for use. It's related to is the server providing the service to the users in displaying images. As our system will be used by thousands of users at any time our system must be available always. If there are any cases of updating, they must be performed in a short interval of time without interrupting the normal services made available to the users.

Efficiency: Specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth etc. All of the above-mentioned resources can be effectively used by performing most of the validations at client side and reducing the workload on server by using JSP instead of CGI which is being implemented now.

Flexibility: If the organization intends to increase or extend the functionality of the software after it is deployed, that should be planned from the beginning; it influences choices made during the design, development, testing and deployment of the system. New modules can be easily integrated to our system without disturbing the existing modules or modifying the logical database schema of the existing applications.

Portability: Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run. By using appropriate server versions released for different platforms our project can be easily operated on any operating system, hence can be said highly portable.

Scalability: Software that is scalable has the ability to handle a wide variety of system configuration sizes. The non-functional requirements should specify the ways in which the system may be expected to scale up (by increasing hardware capacity, adding machines etc.). Our system can be easily expandable. Any additional requirements such as hardware or software which increase the performance of the system can be easily added. An additional server would be useful to speed up the application.

Integrity: Integrity requirements define the security attributes of the system, restricting access to features or data to certain users and protecting the privacy of data entered into the software. Certain features access must be disabled to normal users such as adding the details of files, searching etc which is the sole responsibility of the server. Access can be disabled by providing appropriate logins to the users for only access.

Usability: Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and used by its intended users. Hyperlinks will be provided for each and every service the system provides through which navigation will be easier. A system that has high usability coefficient makes the work of the user easier.

Performance: The performance constraints specify the timing characteristics of the software

.

3.7 System Requirements

3.7.1 Software Requirements

- Operating System: Windows 8 and above
- Coding Language: Python
- Libraries: OpenCV, matplotlib

3.7.2 Hardware Requirements

- Processor: Core i3 and above
- RAM: 16 GB
- Clock Speed: 3.6 GHz
- SSD: 1 TB

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 System Architecture Design:

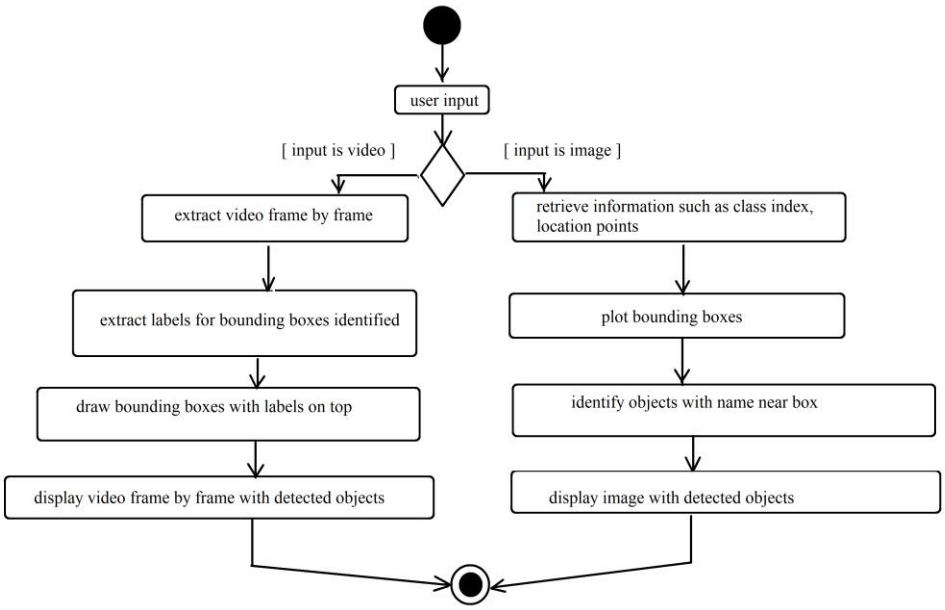
A system Architecture is the conceptual model that defines the structure, behavior, and more views of the system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can consist of system components and the sub-systems developed, that will work together the overall system.

The general flow of our object detection algorithm is fairly straight forward.

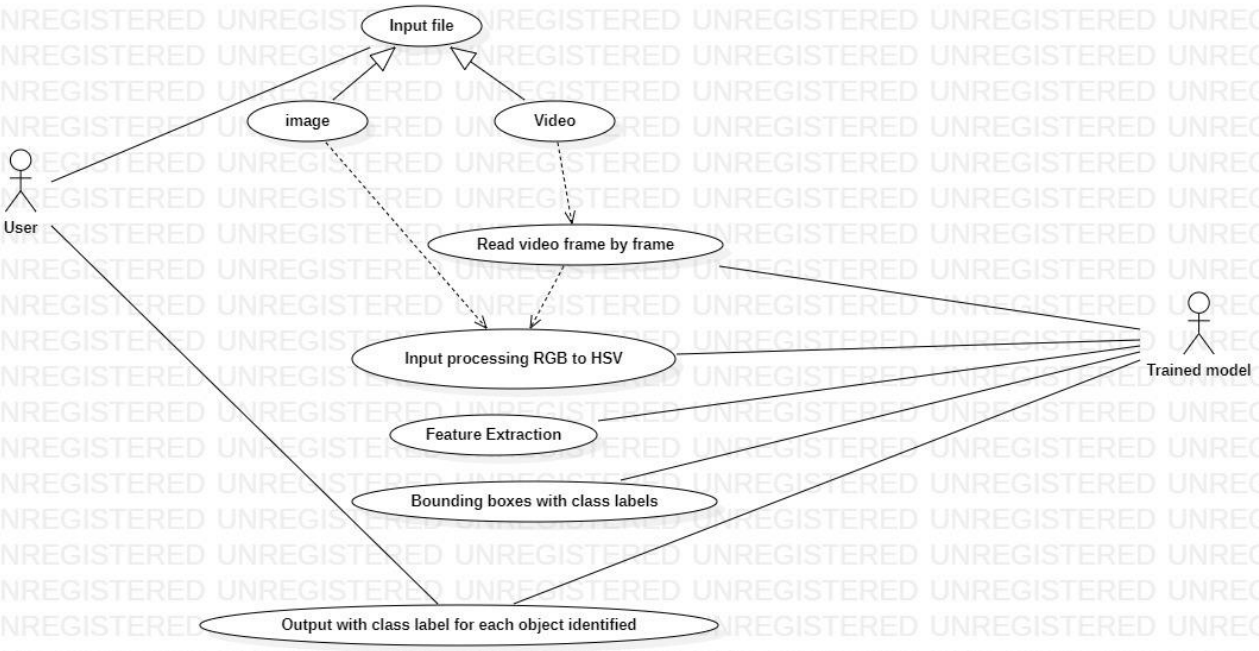
1. First, an input data in either image or video format is given.
2. Once input data is found, it is processed by Open CV, then it is fed to algorithms used such as YOLO and Mobile Netv2,v3.
3. Then the data set is used compute the confidence value to determine the object and also some of the objects in background of the input data given.

Object Diagram

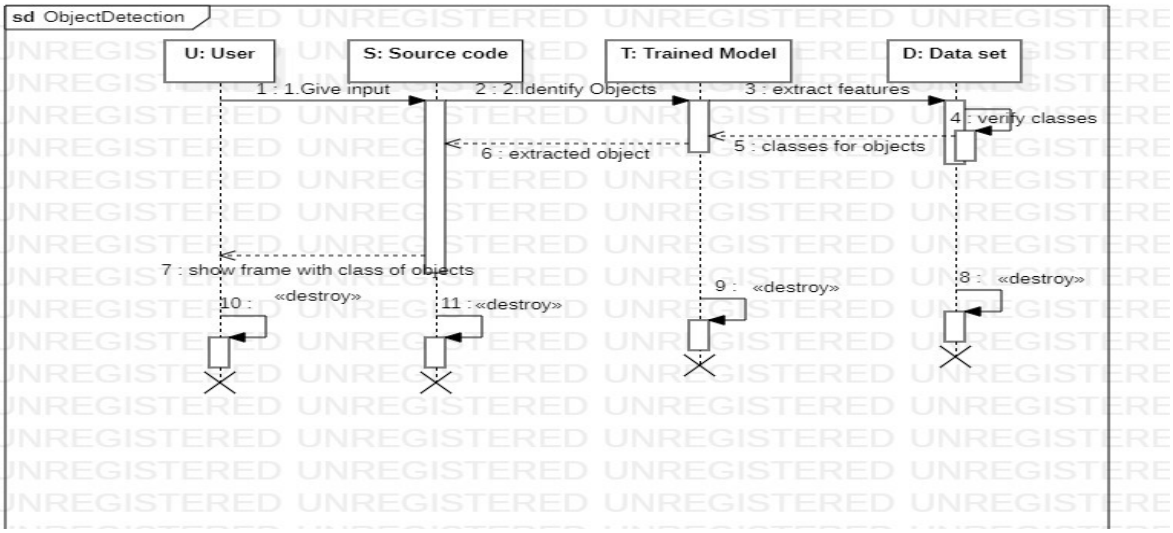
Activity Diagram



Use case diagram



Sequence Diagram



CHAPTER 5

TECHNOLOGIES

5. Technologies

5.1 Python

5.1.1 About Python

Python is currently the most widely used multi-purpose, high-level programming language. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

The biggest strength of Python is huge collection of standard library which can beused for the following

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQtetc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks

Advantages of Python:

- **Extensive Libraries:** Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

- **Extensible:** As we have seen earlier, Python can be extended to other languages. Youcan write some of your code in languages like C++ or C. This comes in handy, especially in projects.

- **Embeddable**: Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

- **Improved Productivity**: The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

- **Simple and Easy:** When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pickup Python, they have a hard time adjusting to other more verbose languages like Java.

- **Readable:** Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

- **Object-Oriented:** This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

- **Free and Open-Source:** Like we said earlier, Python is freely available. But not onlycan you download python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

- **Portable**: When you code your project in a language like C++, you may need to makesome changes to it if you want to run it on another platform. But it isn't the same withPython. Here, you need to code only once, and you can run it any where. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

- **Interpreted:** Lastly, we will say that it is an interpreted language. Since statements areexecuted one by one, debugging is easier than in compiled languages.

Disadvantages of Python:

- **Speed Limitations:** We have seen that Python code is executed line by line. But since python is interpreted, it often results in **slow execution.**

- This, however, isn't a problemunless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

- **Weak in Mobile Computing and Browsers**: While it serves as an excellent server-sidelanguage, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smart phone-based applications. One such application is called Carbonnelle.

- **Design Restrictions**: As you know, Python is dynamically-typed. This means that youdon't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck.While this is easy on the programmers during coding, it can raise run-time errors.

- **Under developed Database Access Layers:** Compared to more widely used technologies like JDBC (Java Data Base Connectivity) and ODBC (Open Data Base Connectivity), Python's data base access layers are a bit under developed. Consequently, it is less often applied in huge enterprises.

5.1.2 Required Python Libraries:

- **OpenCV:** OpenCV is the huge open-source library. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

- **Matplotlib**: Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. Matplotlib can be used in multiple ways including Python scripts, the Python and iPython shells, Jupyter Notebooks. Matplotlib is a 2-D plotting library.

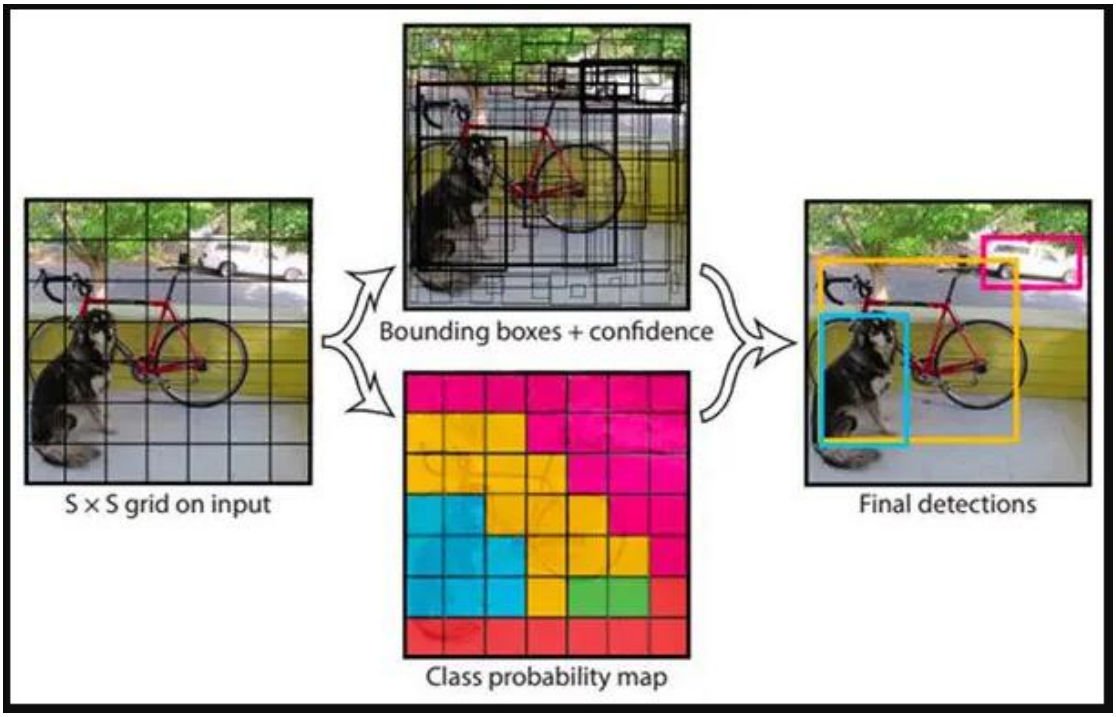
5.2 You Only Look Once (YOLO)

YOLO algorithm is **an algorithm based on regression**, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in one run of the Algorithm. YOLO trains on **full images** and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

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.

Object detection is one of the classical problems in computer vision where you work to recognize *what* and *where* — specifically what objects are inside a given image and also where they are in the image. The problem of object detection is more complex than classification, which also can recognize objects but doesn't indicate where the object is located in the image. In addition, classification doesn't work on images containing more than one object.

YOLO algorithm works using the following three techniques:
Residual blocks: First, the image is divided into various grids. Each grid has a dimension of S x S. Every grid cell will detect objects that appear within them.
Bounding box regression: A bounding box is an outline that highlights an object in an image.
Every bounding box in the image consists of 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)



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.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.

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.
- YOLO provides a high speed of 45 FPS for large networks and 150 FPS for smaller ones. Moreover, YOLO generalizes the image and does not take a toll on processing memory.

Disadvantages of YOLO:

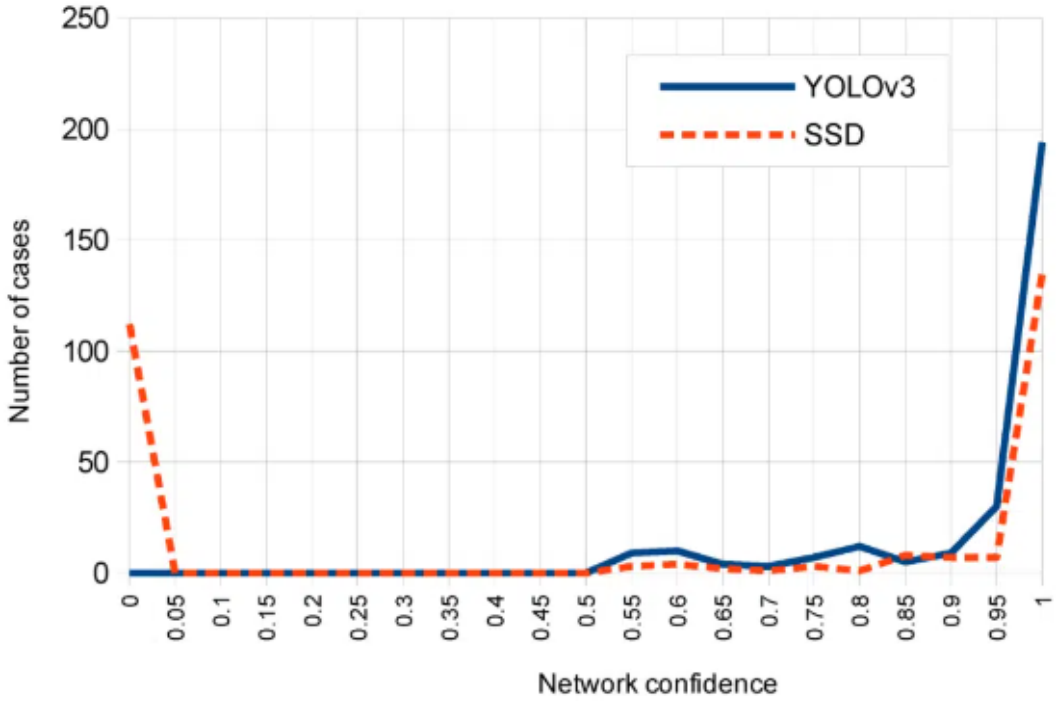
- Comparatively low recall and more localization error compared to Faster R_CNN.
- Struggles to detect close objects because each grid can propose only 2 bounding boxes. Struggles to detect small objects

.5.3 MobileNet SSD

MobilenetSSD is **an object detection model that computes the bounding box and category of an object from an input image**. This Single Shot Detector (SSD) object detection model uses Mobilenet as backbone and can achieve fast object detection optimized for mobile devices.

YOLO (You Only Look Once) is an open-source object detection system. It can recognize objects on a single image or a video stream rapidly. SSD (Single-Shot Multi-box Detection) detects objects with high precision in a single forward pass computing feature map. It can work on video live-streams with a discreet exactness trade-off.

Both YOLO and SSD are top object detection tools, one quicker and the other more precise.



CHAPTER 6

IMPLEMENTATION

6. Implementation

Firstly, required libraries openCV, matplolib are imported.

```
In [2]: import cv2

In [3]: import matplotlib.pyplot as plt
```

We need the configuration file to train the model and frozen model file. An object detection model is trained to detect the presence and location of multiple classes of objects. For example, a model might be trained with images that contain various pieces of fruit, along with a label that specifies the class of fruit they represent (e.g. an apple, a banana, or a strawberry), and data specifying where each object appears in the image.

```
In [4]: config_file = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
        frozen_model = 'frozen_inference_graph.pb'

In [5]: model = cv2.dnn_DetectionModel(config_file,frozen_model)
```

Then we read the coco dataset which is having 80 classes.

```
In [5]: model = cv2.dnn_DetectionModel(config_file,frozen_model)

In [6]: classLabels = []
        file_name = 'Labels.txt'
        with open(file_name,'rt') as fpt:
            classLabels = fpt.read().rstrip('\n').split('\n')

In [7]: print(classLabels)
```

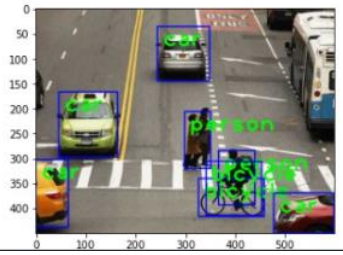
Here, set the various parameters such as size, scaling, mean and color values of the input image. So, the parameters provided to the model will act as training parameters. We proceed to reading input file from user.

From the image with the help of our trained model we will retrive information such as class index (object), confidence (accuracy level) and bbox (location co-ordinates).

It's the time to plot the boxes around the object and identify the object by printing its name near the box. As a result, we will set the values for font, font scale, color of rectangle, font color, thickness of box and display the image with the detected objects.

```
In [20]: ClassIndex,confidece,bbox=model.detect(img,confThreshold=0.5)
         print(ClassIndex)
         font_scale = 3
         font =cv2.FONT_HERSHEY_PLAIN
         for ClassInd, conf,boxes in zip(ClassIndex.flatten(),confidece.flatten(),bbox):
             cv2.rectangle(img,boxes,(255,0,0),2)
             cv2.putText(img,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40),font,fontScale=font_scale,color=(0,255,0),thickness=3)
             plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))

         [3 2 1 1 3 3 3 2]
```



Incase of input is video format also we follow the same procedure as illustrated above, however, few changes included as follows for processing video input.

Create an instance of **VideoCapture** with argument the name of a video file. Pass q as the device index for the camera.

Once the instance of VideoCapture is created, you can capture the video frame-by-frame.

We will first check if the VideoCapture was initialized or not using **cap.isOpened()**. If cap.isOpened() is not True then open it using **cap.open()**.

We set up an infinite loop and read the video frame by frame using cap.read(). cap.read() returns Boolean. If the frame is read correctly cap.read(0 return True.

We extract the top left, bottom right and the label to draw the bounding box with the label displayed on top of the bounding box.

We show the frame in a window using cv2.imshow().

To exit and release the capture, press q.

In [ ]:

```
import cv2
import matplotlib.pyplot as plt
config_file = './ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
frozen_model = './frozen_inference_graph.pb'
model = cv2.dnn_DetectionModel(config_file,frozen_model)
model.setInputSize(320,320)
model.setInputScale(1.0/127.5)
model.setInputMean((127.5,127.5,127.5))
model.setInputSwapRB(True)
classLabels = []
file_name = 'Labels.txt'
with open(file_name,'rt') as fpt:
    classLabels = fpt.read().rstrip('\n').split('\n')

cap=cv2.VideoCapture("side.mp4")
if not cap.isOpened():
    cap =cv2.VideoCapture("")
if not cap.isOpened():
    raise IOError("Cannot open video")

font_scale = 3
font=cv2.FONT_HERSHEY_PLAIN
```

```
while True:
    ret,frame = cap.read()

    ClassIndex, confidece, bbox = model.detect(frame,confThreshold=0.55)

#     print(ClassIndex)
    if (len(ClassIndex)!=0):
        for ClassInd,conf,boxes in zip(ClassIndex.flatten(),confidece.flatten(),bbox):
            if(ClassInd<=80):
                cv2.rectangle(frame,boxes,(255,0,0), 2 )
                cv2.putText(frame,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40),font,fontScale=font_scale,color=(0,255,0),thi
    cv2.imshow('Object Detection Tutorial',frame)
    if cv2.waitKey(2) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

CHAPTER 7

TESTING

7. Testing

7.1 Purpose of Testing

- The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring it.

- Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specifictesting requirement.

7.2 Types of Testing

7.2.1 Unit testing

- Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.

- It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2.2 Integration testing

• Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 Functional testing

• Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

• Functional testing is centered on the following items:

  • Valid Input: identified classes of valid input must be accepted.

  • Invalid Input: identified classes of invalid input must be rejected.

  • Functions: identified functions must be exercised.

  • Output: identified classes of application outputs must be exercised.

  • Systems/Procedures: interfacing systems or procedures must be invoked.

• Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.2.4 System Testing

• System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results.

An example of system testing is the configuration-oriented system integration test. System testing is based onprocess descriptions and flows, emphasizing pre-driven process links and integration points.

7.2.5 White Box Testing

- White Box Testing is a testing in which in which the software tester has knowledge ofthe inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.2.6 Black Box Testing

- Black Box Testing is testing the software without any knowledge of the inner workings,structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

- It is a testing in which the software under test is treated, as a black box. you cannot see into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

- Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.3 Test Strategy and Design

Designing a test strategy for all different types of functioning, hardware by determining the efforts and costs incurred to achieve the objectives of the system. For any project, the test strategy can be prepared by

- Defining the scope of the testing.
- Identifying the type of testing required
- Risks and issues
- Creating test logistics.

7.4 Test Objectives

Test objective is the overall goal and achievement of the test execution. Objectives are defined in such a way that the system is bug-free and is ready to use bythe end-users. Test objective can be defined by identifying the software features that are needed to test and the goal of the test, these features need to achieve to be notedas successful.

7.5 Features to be Tested:

- Object extraction from input image / video.
- Number of objects detected and number of objects correctly detected.

7.6 Test Cases:

TEST CASES	**NO.OF OBJECTS**	**DETECTED OBJECTS**	**CORRECTLY DETECTED**	**WRONGLY DETECTED**	**NOT DETECTED**
CASE 1	23	21	21	-	2
CASE 2	8	8	8	-	-
CASE 3	1	1	1	-	-
CASE 4	30	21	21	1	9
CASE 5	18	14	-	-	4
CASE 6	9	9	9	-	-

CHAPTER 8

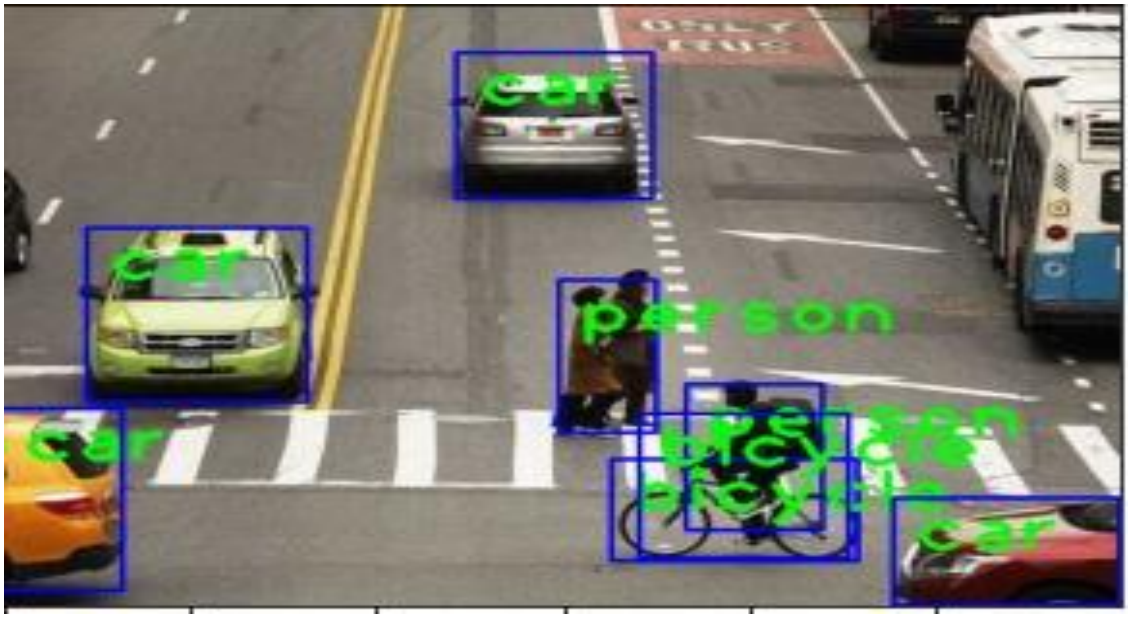
SCREENSHOTS

Input -1:

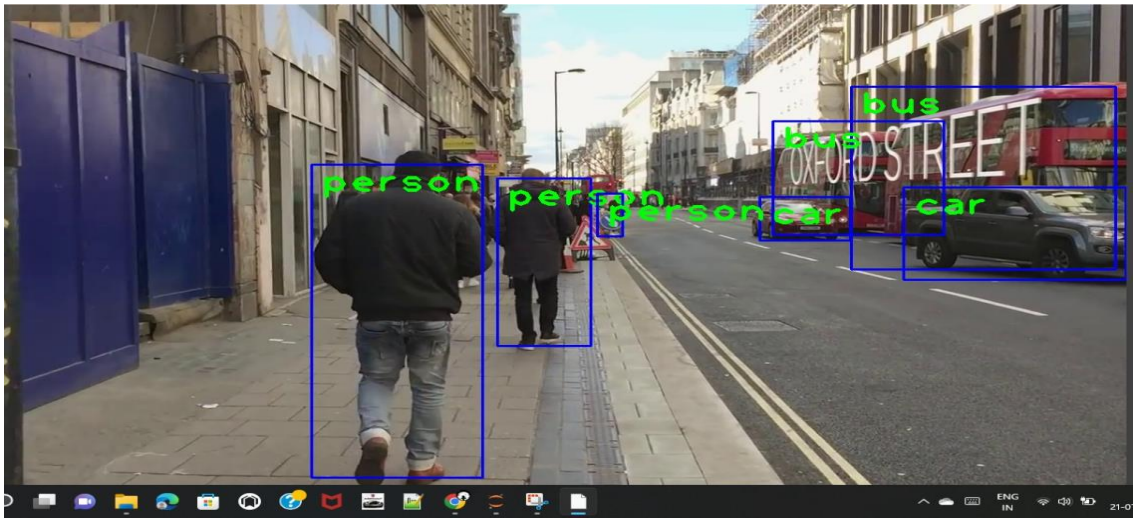
After detecting objects:

Input -2:

After detecting objects:



Object detection in video: (a frame is shown here)



CHAPTER 9

CONCLUSION AND FUTURE WORK

9. Conclusion and Future Work

Conclusion:

The road accidents have proved to be a menace that has majorly reduced the safety of the general public, let alone the driver. The World Health Organization identified sleepiness, alcoholism, and carelessness as the significant causes of road accidents in their Global Status Report on Road Safety. As a result, the fatalities and associated expenses that follow prove to be a severe threat to families across the world.

Therefore, in this study, a low-cost, real-time object detection system is developed with acceptable accuracy in detecting obstacles. In the developed system, input in image or video format is taken and objects detected in each frame deploying image processing techniques. Bounding boxes on the detected objects are intersected to eliminate unnecessary objects and objects identified are predicted based on pre-defined threshold.

This system can be elaborated by training on more datasets to improve the efficiency of output. This project can be integrated with camera detection to customize object detection by dynamically imparting input to the software version developed so far.

With the extension feature added through camera detection, the system could work much better in real-time environment.

REFERENCES

REFERENCES

[1] Object Detection for Autonomous Vehicles Gene Lewis Stanford University Stanford, CA.

[2] Chengji Liu ,Yufan Tao ,Jiawei Liang ,Kai Li1 ,Yihang Chen, ”Object Detection Based on YOLO Network” 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC 2018)

[3] Rumin Zhang, Yifeng Yang, ”An Algorithm for Obstacle Detection based on YOLO and Light Filed Camera”, 2018 Twelfth International Conference on Sensing Technology (ICST)

[4] Joseph Redmond, Ali Farhadi, ”YOLOv3: An Incremental Improvement”, University of Washington

[5] Joseph Redmon,Santosh Divvala,Ross Girshick,Ali Farhadi. ”You Only Look Once: Unified, Real-Time Object Detection” [J].2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2016:779788

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In Conference on Computer Vision and Pattern Recognition (CVPR), 2012.