Project Report

on

Image to a pencil sketch

Submitted By

Atik Ahmed - 2100290140041
Harshit Kaushik - 2100290140068
Mohd Wasim - 2100290140085

Submitted in partial fulfilment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Dr. Vidushi

Submitted to

Department of Computer Applications,
KIET Group of Institutions,
Delhi-NCR, Ghaziabad
Uttar Pradesh – 201206

(June 2023)

DECLARATION

I hereby declare that the work presented in report entitled “Image to a pencil sketch” was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University of Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, that are not my original contribution. I have used quotation marks to identify verbatim sentences and give credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Atik Ahmed
Roll No.: 2100290140041

(Candidate Signature)

Name: Harshit Kaushik
Roll No.: 2100290140068

(Candidate Signature)

Name: Mohd Wasim
Roll No.: 2100290140085

(Candidate Signature)

CERTIFICATE

Certified that **Atik Ahmed(2100290140041), Harshit Kaushik(2100290140068), Mohd Wasim(2100290140085)** have carried out the project work **"Image to a pencil sketch"**for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the students themselves and the contents of the project report do not form the basis for the award of any other degree to the candidates or to anybody else from this or any other University/Institution.

Atik Ahmed (2100290140041)
Harshit Kaushik (2100290140068)
Mohd Wasim (2100290140085)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vidushi

Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Arun Kumar Tripathi
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

This project is about how new technologies can be used to develop a python application that enables the user to make a sketch of any Images. We will try to convert a normal photo into a sketch using computer vision, machine learning in a python programming Language. We also try enhance by adding image detection. In image detection you can easily detect the type of image it is. We can easily classified whose image is show.

This Project make it easier for people to understand how painters create pencil drawings, one of the most basic pictorial languages for representing the abstract perception of natural scenes.

This is accomplished through the use of graphical interface Python tools such as Tkinter, matplotlib, and others. The user can easily choose an image from their files. Then it can select the convert button. The picture has been transformed into a lovely sketch.

The project's primary goal is to better human-computer interaction by creating a method to assist people by converting their various images into a beautiful sketch. The user is provided with a decent user interface to upload a sketch input image and obtain related output images.

Image detection is the method of classifying the image with the trained model when given with the image that is to be classified by extracting feature from the image. With the advancement in the field of machine learning it has become easy to train the model to classify various types of images and image detection has gained huge application in real life. This paper presents a study on the image detection using CNN and TenserFlow for the image detection. In this work experiment is performed using different self collected dataset and their results are evaluated.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Vidushi** for his guidance, help, and encouragement throughout my researchwork. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications**, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion ofthis work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Atik Ahmed 2100290140041
HarshitKaushik 2100290140068
Mohd Wasim 2100290140085

Contents

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPTION

In today's era, we are surrounded by different types of photo manipulating filters in our mobile phones, apps…etc. But do you know how they do these images manipulations…..? In the backend, they are using computer vision techniques. Computer vision has a wide variety of applications not only to reduce the human effort but also used for entertainment apps. Many photo editing apps like FaceApp, Instagram filters…etc are using computer vision techniques.

In this Project, we will try to convert a normal photo into a pencil sketch using computer vision in a python programming language. In this Project, we will show how to convert an image into its corresponding pencil sketch in a few steps.This system helps people better understand how painters produce pencil drawings, which is one of the most fundamental pictorial languages to abstract human perception of natural scenes.

When given an image that needs to be classified, image detection uses a trained model to extract features from the image and classify it. With the development of machine learning, it is now simple to train the model to categorise different kinds of photos, and image detection has many practical applications. This article describes a study on image detection that makes use of CNN and TenserFlow. In this study, experiments are carried out utilising various self-collected datasets, and the outcomes are assessed.

TECHNOLOGY USED

Python

Python is a high-level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in the easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages. Python

provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, Data science, etc. Python has a lot of libraries for every need of this project such as Pytube for downloading videos, selenium for web automation, etc. Python is reasonably efficient. Efficiency is usually not a problem for small examples. If your Python code is not efficient enough, a general procedure to improve it is to find out what is taking most of the time and implement just that part more efficiently in some lower-level languages.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

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.

PixelLib

PixelLib is a library used for easy implementation of semantic and instance segmentation of objects in images and videos with few lines of code. PixelLib makes it possible to train a custom segmentation model using few lines of code. PixelLib supports background editing of images and videos using few lines of code.

Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task.

ImageAI:

Image processing is the analysis and manipulation of a digitized image, often to improve its quality. By leveraging machine learning, Artificial intelligence (AI) processes an image, improving the quality of an image based on the algorithm's “experience” or depth of knowledge.

APPLICATION OF OBJECT DETECTION

The major applications of Object Detection are:

FACIAL RECOGNITION "Deep Face" is a deep learning facial recognition system developed to identify human faces in a digital image. Designed and developed by a group of researchers in Facebook. Google also has its own facial recognition system in Google Photos, which automatically separates all the photos according to the person in the image. There are various components involved in Facial Recognition or authors could say it focuses on various aspects like the eyes, nose, mouth and the eyebrows for recognizing a faces.

PEOPLE COUNTING People counting is also a part of object detection which can be used for various purposes like finding person or a criminal; it is used for analysing store performance or statistics of crowd during festivals. This process is considered a difficult one as people move out of the frame quickly.

INDUSTRIAL QUALITY CHECK Object detection also plays an important role in industrial processes to identify or recognize products. Finding a particular object through visual examination could be a basic task that's involved in multiple industrial processes like sorting, inventory management, machining, quality management, packaging and so on. Inventory management can be terribly tough as things are hard to trace in real time. Automatic object counting and localization permits improving inventory accuracy.

SELF DRIVING CARS Self-driving is the future most promising technology to be used, but the working behind can be very complex as it combines a variety of techniques to perceive their surroundings, including radar, laser light, GPS, odometer, and computer vision. Advanced control systems interpret sensory info to allow navigation methods to work, as well as obstacles and it. This is a big step towards Driverless cars as it happens at very fast speed.

SECURITY Object Detection plays a vital role in the field of Security; it takes part in major fields such as face ID of Apple or the retina scan used in all the sci-fi movies. Government also widely use this application to access the security feed and match it with their existing database to find any criminals or to detecting objects like car number involved in criminal activities. The applications are limitless.

YOLO(You Only Look Once) YOLO which stands for "You only look once" is a single shot detection algorithm which was introduced by Joseph Redmon in May 2016. Although the name of the algorithm may sound strange, it gives a perfect description of this algorithm as it predicts classes and bounding boxes for the whole image in one run of the algorithm.

YOLO performed surprisingly well as compared to the other single-shot detectors of that time in terms of speed and accuracy. It is not the most accurate algorithms when it comes to object detection but certainly, it makes that up with its impressive speed and thus is a good balance between speed and accuracy.

What Makes YOLO Popular for Object Detection?

Some of the reasons why YOLO is leading the competition include its:

- Speed
- Detection accuracy
- Good generalization
- Open-source

1- Speed

YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing.

From the graphic below, we observe that YOLO is far beyond the other object detectors with 91 FPS.

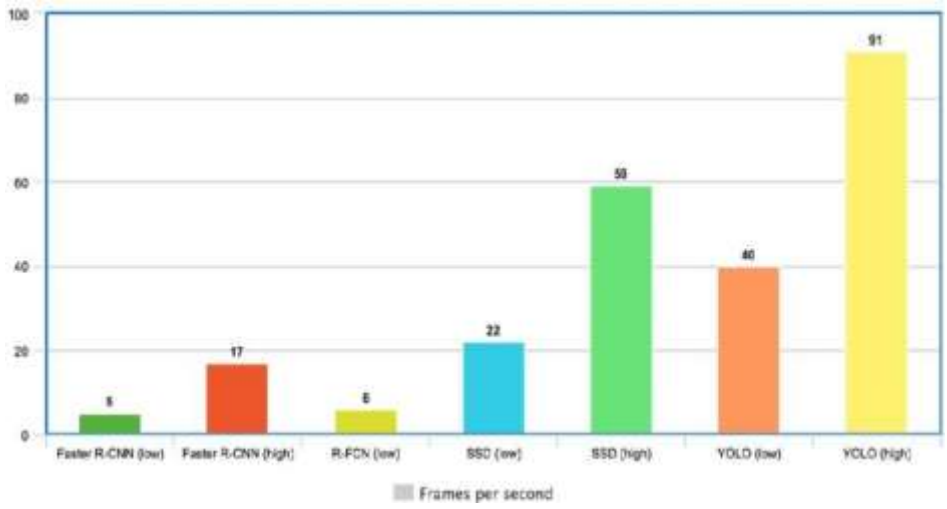

Fig: 1.1 YOLO Speed compared to other state-of-the-art object detectors.

High detection accuracy

YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.

Better generalization

This is especially true for the new versions of YOLO, which will be discussed later in the article. With those advancements, YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection.

For instance the Automatic Detection of Melanoma with Yolo Deep Convolutional Neural Networks paper shows that the first version YOLOv1 has the lowest mean average precision for the automatic detection of melanoma disease, compared to YOLOv2 and YOLOv3.

Open source

Making YOLO open-source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

YOLO Architecture

YOLO architecture is similar to **GoogleNet**. As illustrated below, it has overall 24 convolutional layers, four max-pooling layers, and two fully connected layers.

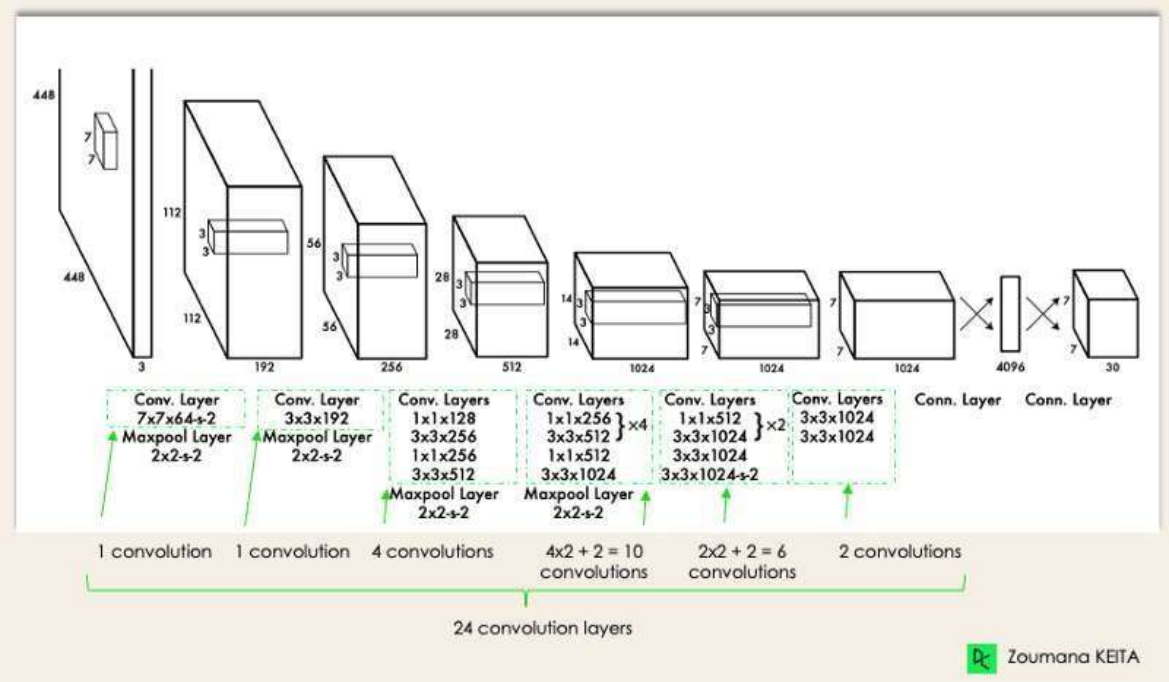

Fig: 1.2 YOLO Architecture

The architecture works as follows:

- Resizes the input image into 448x448 before going through the convolutional network.
- A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal output.
- The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.
- Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

By completing the **Deep Learning in Python** course, you will be ready to use Keras to train and test complex, multi-output networks and dive deeper into deep learning.

How Does YOLO Object Detection Work?

Now that you understand the architecture, let's have a high-level overview of how the YOLO algorithm performs object detection using a simple use case.

"Imagine you built a YOLO application that detects players and soccer balls from a given image.

But how can you explain this process to someone, especially non-initiated people?

→ That is the whole point of this section. You will understand the whole process of how YOLO performs object detection; how to get image (B) from image (A)"

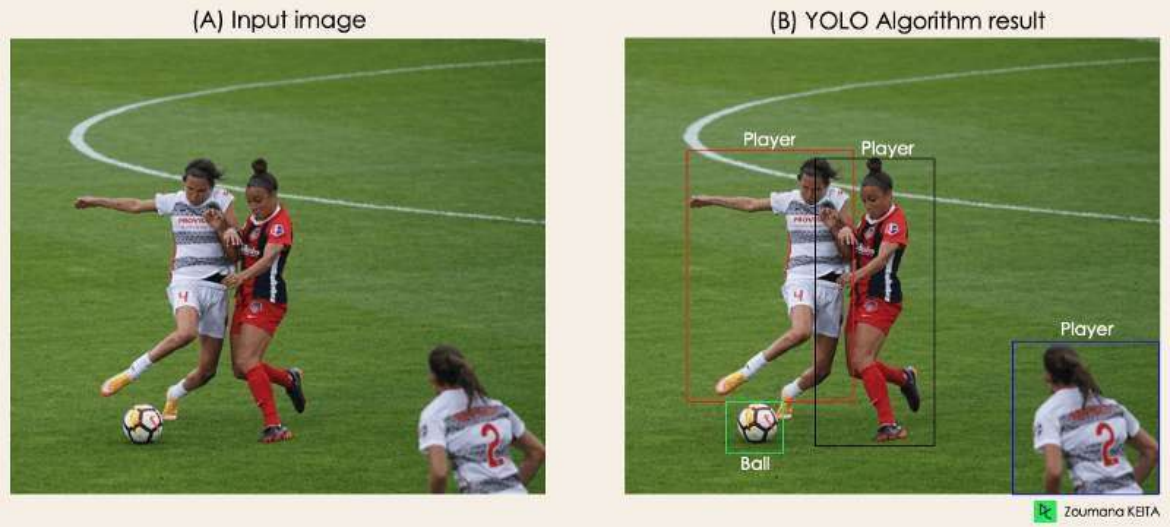

Fig:1.3

The algorithm works based on the following four approaches:

- Residual blocks
- Bounding box regression
- Intersection Over Unions or IOU for short
- Non-Maximum Suppression.

Let's have a closer look at each one of them.

Residual blocks

This first step starts by dividing the original image (A) into NxN grid cells of equal shape, where N in our case is 4 shown on the image on the right. Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, along with the probability/confidence value.

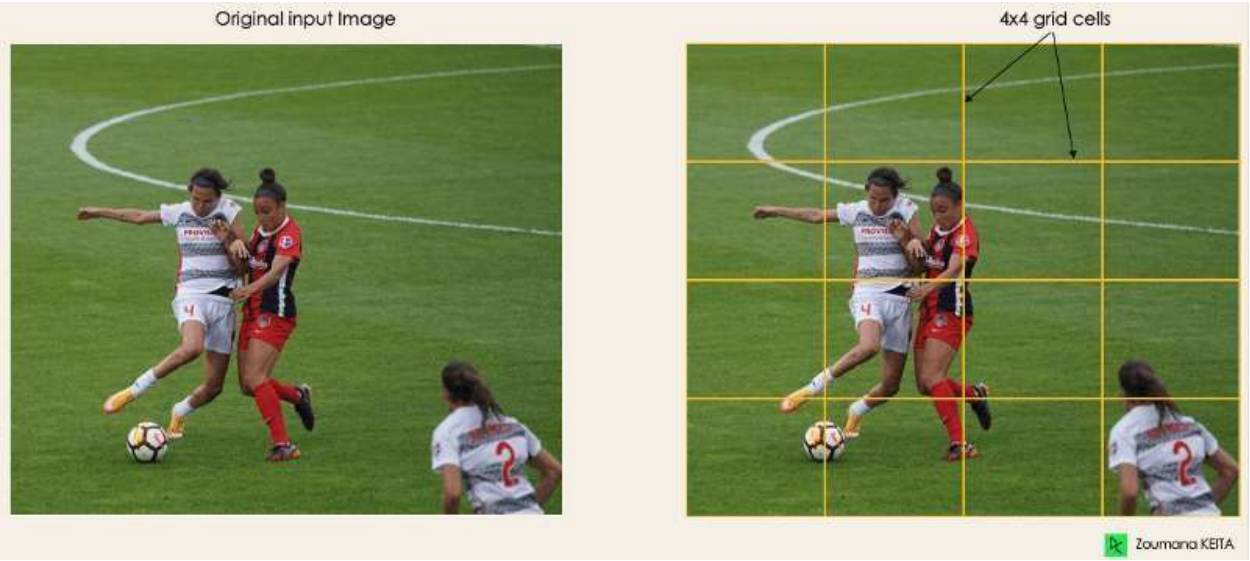

Fig: 1.4

Bounding box regression

The next step is to determine the bounding boxes which correspond to rectangles highlighting all the objects in the image. We can have as many bounding boxes as there are objects within a given image.

YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

Y = [pc, bx, by, bh, bw, c1, c2]

This is especially important during the training phase of the model.

- pc corresponds to the probability score of the grid containing an object. For instance, all the grids in red will have a probability score higher than zero. The image on the right is the simplified version since the probability of each yellow cell is zero (insignificant).

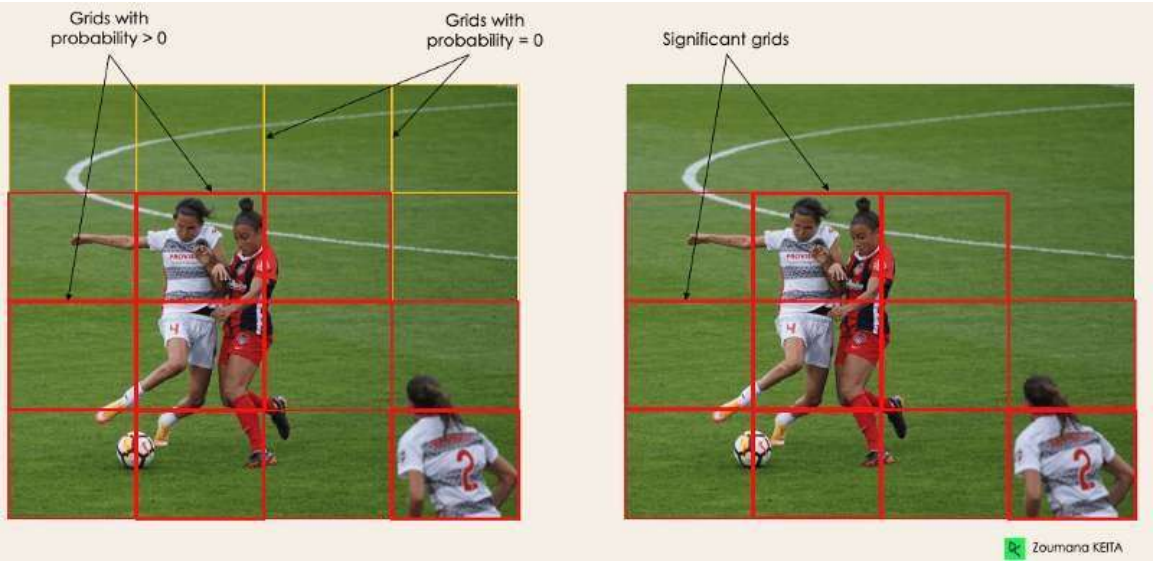

Fig: 1.5

- bx, by are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell.
- bh, bw correspond to the height and the width of the bounding box with respect to the enveloping grid cell.
- c1 and c2 correspond to the two classes Player and Ball. We can have as many classes as your use case requires.

  To understand, let's pay closer attention to the player on the bottom right.

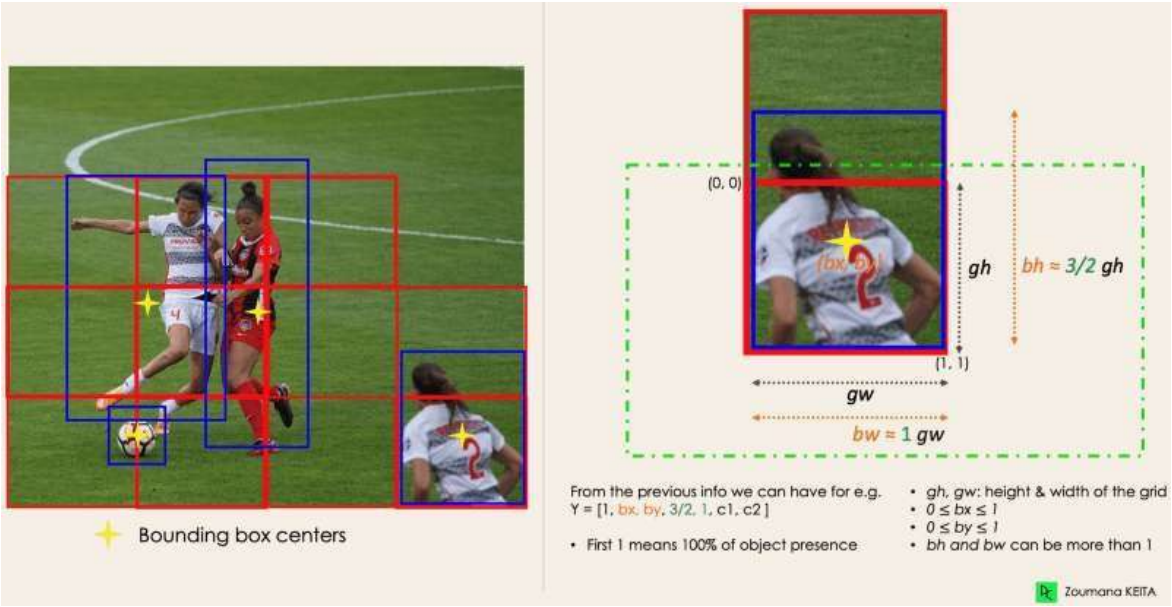

Fig: 1.5

Intersection Over Unions or IOU

Most of the time, a single object in an image can have multiple grid box candidates for prediction, even though not all of them are relevant. The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant. Here is the logic behind it:

- The user defines its IOU selection threshold, which can be, for instance, 0.5.
- Then YOLO computes the IOU of each grid cell which is the Intersection area divided by the Union Area.
- Finally, it ignores the prediction of the grid cells having an IOU ≤ threshold and considers those with an IOU > threshold.

Below is an illustration of applying the grid selection process to the bottom left object. We can observe that the object originally had two grid candidates, then only "Grid 2" was selected at the end.

Fig: 1.6

Non-Max Suppression or NMS

Setting a threshold for the IOU is not always enough because an object can have multiple boxes with IOU beyond the threshold, and leaving all those boxes might include noise. Here is where we can use NMS to keep only the boxes with the highest probability score of detection.

YOLO Applications

YOLO object detection has different applications in our day-to-day life. In this section, we will cover some of them in the following domains: healthcare, agriculture, security surveillance, and self-driving cars.

Application in industries

Object detection has been introduced in many practical industries such as healthcare and agriculture. Let's understand each one with specific examples.

Healthcare

Specifically in surgery, it can be challenging to localize organs in real-time, due to biological diversity from one patient to another. **Kidney Recognition in CT used YOLOv3** to facilitate localizing kidneys in 2D and 3D from computerized tomography (CT) scans.

The **Biomedical Image Analysis in Python course** can help you learn the fundamentals of exploring, manipulating, and measuring biomedical image data using Python.

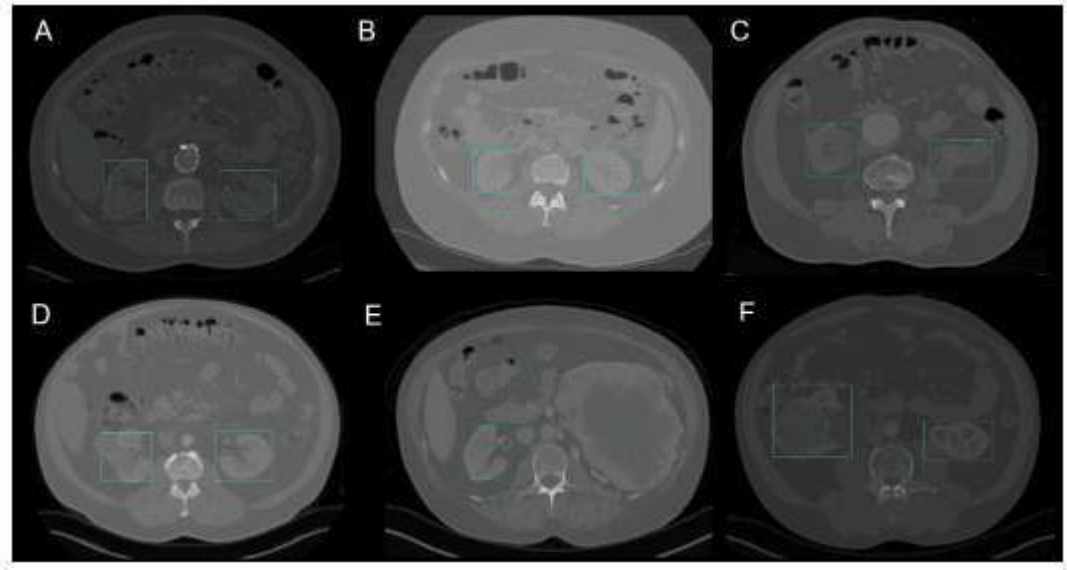

Fig:1.6 2D Kidney detection by YOLOv3 (Image from **Kidney Recognition in CT usingYOLOv3**)

Agriculture

Artificial Intelligence and robotics are playing a major role in modern agriculture. Harvesting robots are vision-based robots that were introduced to replace the manual picking of fruits and vegetables. One of the best models in this field uses YOLO.

In **Tomato detection based on modified YOLOv3 framework**, the authors describe how they used YOLO to identify the types of fruits and vegetables for efficient harvest.

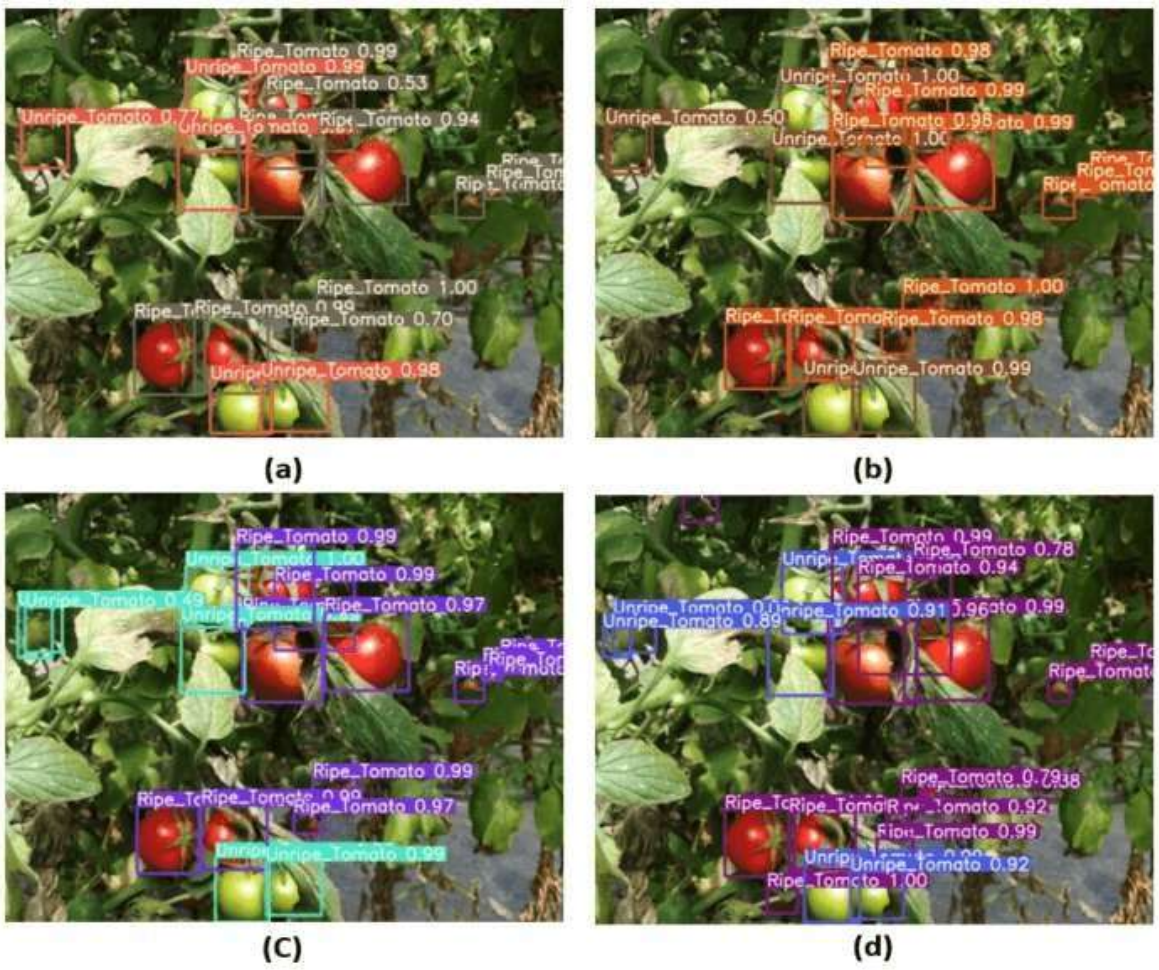

Fig: 1.7 Image from Tomato detection based on modified YOLOv3 framework

Security surveillance

Even though object detection is mostly used in security surveillance, this is not the only application. YOLOv3 has been used during covid19 pandemic to estimate social distance violations between people.

Self-driving cars

Real-time object detection is part of the DNA of autonomous vehicle systems. This integration is vital for autonomous vehicles because they need to properly identify the correct lanes and all the surrounding objects and pedestrians to increase road safety. The real-time aspect of YOLO makes it a better candidate compared to simple image segmentation approaches.

OBJECT DETECTION WORKFLOW AND FEATURE EXTRACTION

Every Object Detection Algorithm works on the same principle and it's just the working that differs from others. They focus on extracting features from the images that are given as the input at hands and then it uses these features to determine the class of the image.

LITERATURE REVIEW

Among the many uses for object detection, security is one of the most crucial ones. Of course, there are many applications of object detection utilizing machine learning in the r ealm of security. The topic that we opted to conduct a literature review on has not yet been c overed by a publication, as far as we can tell. And to our knowledge, no one has conducted a thorough literature study on the subject we have chosen. The two studies that are the close st to being a survey that we have discovered are either specific or cover particular methodol ogies rather than analyze other papers.

A. A study on several techniques for object detection and tracking in video surveilla nce footage was done in one work by Murugan et al. [1]. The paper highlights how video su rveillance has existed as a technology since the 1950s and reinforces the point stated in the I ntroduction section about how watching security cameras can be tiresome and have an adve rse effect on a person's mental health. It also illustrates how automating the procedure was t he answer to that tedious chore by introducing a sophisticated surveillance system. This pap er's main goal is to describe the various techniques for object detection and tracking in vide os. The first technique described in the study is backdrop subtraction. The report claims that one of the most popular techniques for spotting moving items is backdrop removal. To show only the moving object's pixels, background subtraction includes identifying the backgroun d and removing it. The issue with it is that the results of the procedure are impacted if the ba ckground is not static and changes as a result of illumination or specific weather conditions. There are other algorithms for background subtraction as well.

B. A study by Flitton et al. [2] compares various 3D interest point descriptors for CT images of baggage at airports. Finding interesting stuff during baggage x-ray inspections is t he core concept here. Five distinct methods were compared in the paper: density, density his togram, density gradient histogram, rotation invariant feature transform, and scale invariant feature transform. The report does a decent job of assessing each while providing crucial in dicators, although the research was only done on a relatively narrow issue. Both of the afore mentioned studies do a great job of describing the various strategies, but in comparison to o ur paper, they are too narrow in scope. Instead of separating the focus on machine learning a nd object detection, both publications provide deeper explanation of the object detection tec

hniques. The two articles cover fewer papers than our Systematic Literature Review does be cause of their narrower scopes.

tember - 2016) suggests a deep learning approach for item recognition in historical a rchitecture images in Trnava. For jobs requiring object recognition, it employs deep learnin g architectures based on CNN (Convolutional Neural Network) [3]. Architecture is improve d by using activation functions and a cascade of convolutional layers. Setting up the number of layers and the number of neurons in each layer is crucial. For this, the TRNAVA LeNet 1 0 model was created and trained. This model is based on a dataset of 460 training images an d 140 validation images, or a 3:1 ratio. The images are 28x28 pixels in size, were color phot ographs, and were encoded in jpg format. The model correctly identified the correct item in the picture of the Trnava historical building. The proposed model's prediction accuracy incr eased to 98.88%.

D. Jung, H., Lee, S. et al (Jan – 2015), suggested that instead of using manually prod uced characteristics, deep learning techniques should be used to recognize face expressions. Convolutional neural networks (CNN) and deep neural networks (DNN) are two types of de ep networks that are used to tackle recognizing difficulties [4]. The deep networks were cre ated quickly utilizing deep learning toolkits that enable CUDA, such as Caffe and CudaCon vnet2. Additionally, they made use of the OpenCV library to create the Haar-like face detec tion technique. The photographs were reduced in size and cropped to 64 * 64. The 327 face photos were then separated into ten groups, one of which was utilized for training and the ot her nine for testing. For six emotions, the recognition rates were good, but the disgust label had a low recognition rate. Because there were only 547 training photos for the disgust labe l in the FER 2013 database. Over-fitting is a possibility with the DNN.

E. Tenguria, R., Parkhedkar et al (April – 2017), Convolutional neural networks hav e been replaced by more precise yet sophisticated approaches that can recognize things in re al-time, according to the study [5]. This paper promises a significant advance in object reco gnition and tagging. However, development in this area has been somewhat modest. It inten ds to combine the domains of computer vision and robotics, with a particular emphasis on t he implementation of image description applications on an embedded system platform. A fi xed number of items are allowed in the image, according on the data set used to train the mo

del. Shaoqing Ren et al. claim that the development of the Region Proposal Network (RPN) permits sharing of the entire image's convolutional characteristics with the network, resultin g in nearly free region proposals. In this case, the algorithm is directed by the region sugges tion technique in order to find objects present in the image. Second, the application of this t echnique in our system makes it computationally efficient and tailored to function on low-p owered platforms.

Gao, Q. (Sep. – 2017): In order to improve the effectiveness of current object recog nition algorithms, the research [6] presents an object localization method that makes use of i mage edge information as a cue to pinpoint the locations of the objects. The perceptual orga nisation components of human vision are used to extract the Generic Edge Tokens (GETs) o f the image. To precisely locate objects, these edge tokens are parsed using the Best First Se arch method, with the detection score provided by the Deep Convolutional Neural Network serving as the goal function. The search space is a collection of edge elements whose overla ps with the current candidate object are greater than zero when the BFS is applied to the obj ect localization and its search space. Real-time testing revealed that the model was more eff ective than the RCNN, and there is still room for improvement by enhancing object localiza tion by combining picture edge, color and texture information, and the learnt properties of t he image.

G. Mazumdar, M., Sarasvathi, V. and Kumar, A. (Aug. - 2017), suggested a techniq ue for creating an interactive application to identify things from films; upon user input, it is also capable of identifying the specific object now displayed on the screen [7]. For this chal lenge, which has a 77% accuracy rate, a sequential frame extraction technique for films as w ell as a deep learning strategy utilizing Convolutional and Fully Connected Neural Network s are applied. The work of computer vision is still fairly difficult even when the item is som ewhat warped, translated, rotated, or partially obscured from view and can still be easily spo tted by humans. Taking advantage of the fact that videos are composed of frames synced wi th some playback audio, the analysis of the video can be done in much more detail by looki ng at the objects present in the frame images themselves, running the classifier to obtain pro babilities for various classes, and then classifying the genre as well as identifying any object s in the video. By adding more datasets and optimizing the hardware setup, this model's ope

rational accuracy is increased, enabling faster and more accurate item categorization over a wider variety of classes.

L. Yang, L., Wang, L., & Wu, S. (2018). The presented article [12] suggests Object detection with an image is essentially what object confirmation is. Three steps are involved in conventional object identification algorithms: region selection, feature extraction, and cla ssification. By applying a single deep convolutional neural network to the image, YOLOv2 reframes object detection as a single regression problem that goes straight from image pixel s to bounding box coordinates and class probabilities at the same time. Multi-feature fusion has emerged as a new paradigm in the architecture of deep convolutional neural networks in recent years. Low-layer filters capture the detail texture information of objects, while high-l ayer filters extract the semantic information. The high sensitivity of radar detection and the high object confirmation accuracy are both features of the intelligent radar perimeter securit y system.

Bhumika Gupta (2017) et a;. proposed object detection is a well-known computer technolog y connected with computer vision and image processing that focuses on detecting objects or its instances of a certain class (such as humans, flowers, animals) in digital images and vide os. There are various applications of object detection that have been well researched includi ng face detection, character recognition, and vehicle calculator. Object detection can be use d for various purposes including retrieval and surveillance. In this study, various basic conc epts used in object detection while making use of OpenCV library of python 2.7, improving the efficiency and accuracy of object detection are presented.

Kartik Umesh Sharma (2017) et al, proposed an object detection system finds objects of the real world present either in a digital image or a video, where the object can belong to any cl ass of objects namely humans, cars, etc. In order to detect an object in an image or a video t he system needs to have a few components in order to complete the task of detecting an obj ect, they are a model database, a feature detector, a hypothesiser and a hypothesiser verifier. This paper presents a review of the various techniques that are used to detect an object, loca lise an object, categorise an object, extract features, appearance information, and many mor e, in images and videos. The comments are drawn based on the studied literature and key iss ues are also identified relevant to the object detection. Information about the source codes a

nd online datasets is provided to facilitate the new researcher in object detection area. An id ea about the possible solution for the multi class object detection is also presented. This pap er is suitable for the researchers who are the beginners in this domain.

Mukesh Tiwari (2017) et al. presented object detection and tracking is one of the critical ar eas of research due to routine change in motion of object and variation in scene size, occlusi ons, appearance variations, and ego-motion and illumination changes. Specifically, feature s election is the vital role in object tracking. It is related to many real time applications like ve hicle perception, video surveillance and so on. In order to overcome the issue of detection, t racking related to object movement and appearance. Most of the algorithm focuses on the tr acking algorithm to smoothen the video sequence. On the other hand, few methods use the p rior available information about object shape, color, texture and so on. Tracking algorithm w hich combines above stated parameters of objects is discussed and analyzed in this research. The goal of this paper is to analyze and review the previous approach towards object trackin g and detection using video sequences through different phases. Also, identify the gap and s uggest a new approach to improve the tracking of object over video frame.

Aishwarya Sarkale (2018) et al. proposed humans have a great capability to distinguish obje cts by their vision. But, for machines object detection is an issue. Thus, Neural Networks ha ve been introduced in the field of computer science. Neural Networks are also called as 'Art ificial Neural Networks'. Artificial Neural Networks are computational models of the brain which helps in object detection and recognition. This paper describes and demonstrates the different types of Neural Networks such as ANN, KNN, FASTER R-CNN, 3D-CNN, RNN etc.

Karanbir Chahal (2018) et al. proposed Object detection is the identification of an object in the image along with its localization and classification. It has wide spread applications and is a critical component for vision based software systems. This paper seeks to perform a rigorous survey of modern object detection algorithms that use deep learning. As part of the survey, the topics explored include various algorithms, quality metrics, speed/size trade offs and training methodologies. This paper focuses on the two types of object detection algorithms- the SSD class of single step detectors and the Faster R-CNN class of two step detectors. Techniques to construct detectors that are portable and fast on low powered devices

are also addressed by exploring new light weight convolutional base architectures. Ultimately, a rigorous review of the strengths and weaknesses of each detector leads us to the present state of the art.

Richard Socher (2018) et al. proposed recent advances in 3D sensing technologies make it possible to easily record color and depth images which together can improve object recognition. Most current methods rely on very welldesigned features for this new 3D modality. We introduce a model based on a combination of convolutional and recursive neural networks (CNN and RNN) for learning features and classifying RGB-D images. The CNN layer learns low-level translationally invariant features which are then given as inputs to multiple, fixed-tree RNNs in order to compose higher order features. RNN scan be seen as combining convolution and pooling into one efficient, hierarchical operation. Our main result is that even RNNs with random weights compose powerful features. Our model obtains state of the art performance on a standard RGB-D object data set while being more accurate and faster during training and testing than comparable architectures such as two-layer CNNs.

Yordanka Karayaneva (2018) et al. presented schools in many parts of the world use robots as social peers in order to interact with children and young students for a rich experience. Such use has shown significant enhancement of children's learning. This project uses the humanoid robot NAO which provides object recognition of colours, shapes, typed words, and handwritten digits and operators. The recognition of typed words provides performance of the corresponding movements in the sign language. Five classifiers including neural networks are used for the handwritten recognition of digits and operators. The accuracy of the object recognition algorithms are within the range of 82%-92% when tested on images captured by the robot including the movements which represent words in the sign language. The five classifiers for handwritten recognition produce highly accurate results which are within the range of 87%-98%. This project will serve as a promising provision for an affective touch for children and young students.

CHAPTER 2

PROJECT REQUIREMENTS

HARDWARE AND SOFTWARE REQUIREMENTS

- Processor i5 and above
- 4 GB Ram and above
- Windows 8 and above

TECHNOLOGY REQUIREMENTS

- Python,
- Tkinter
- Machine learning
- OpenCV
- Matplotlib
- PixelLib

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

This chapter describes about the requirements. It specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes overview of dissertation as well as the functional and non-functional requirement of this dissertation. A SRS document describes all data, functional and behavioral requirements of the software under production or development. SRS is a fundamental document, which forms the foundation of the software development process. Its the complete description of the behavior of a system to be developed. It not only lists the requirements of a system but also has a description of its major feature. Requirement Analysis in system engineering and software engineering encompasses those tasks that go into determining the need or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirement Analysis is critical to the success to a development project. Requirement must be documented, measurable, testable, related to in identified business needs or opportunities, and defined to a level of detail sufficient for system design. The SRS functions as a blueprint for completing a project. The SRS is often referred to as the ”parent” document because all subsequent project management documents, such as design specifications, statements of work, software architecture specification, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and non-functional requirements only.

Thus the goal of preparing the SRS document is to

- To facilitate communication between the customer, analyst, system developers, maintainers.
- To serve as a contrast between purchaser and supplier.
- To firm foundation for the design phase.
- Support system testing facilities.

• Support project management and control.
• Controlling the evolution of the system.

3.2 FUNCTIONAL REQUIREMENTS

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:-

• Input test case must not have compilation and runtime errors.
• The application must not stop working when kept running for even a long time.
• The application must function as expected for every set of test cases provided.
• The application should generate the output for given input test case and input parameters.
• The application should generate on-demand services.

3.3 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

• Product Requirements
• Organizational Requirements
• User Requirements
• Basic Operational Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing non-functional requirements is

detailed in the system architecture. Broadly, functional requirements define what a system is supposed to do and non- functional requirements define how a system is supposed to be.

3.3.1 ORGANIZATIONAL REQUIREMENTS

Process Standards: IEEE standards are used to develop the application which is the standard used by the most of the standard software developers all over the world. Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

3.3.2 USER REQUIREMENTS

The user requirements document (URD) or user requirements specification is a document usually used to software engineering that specifies the requirements the user expects from software to be constructed in a software project. Once the required information is completely gathered it is documented in a URD, which is meant to spell out exactly what the software must do and becomes part of the contractual agreement. A customer cannot demand feature not in the URD, whilst the developer cannot claim the product is ready if it does not meet an item of the URD. The URD can be used as a guide to planning cost, timetables, milestones, testing etc. The explicit nature of the URD allows customers to show it to various stakeholders to make sure all necessary features are described. Formulating a URD requires negotiation to determine what is technically and economically feasible. Preparing a URD is one of those skills that lies between a science and economically feasible. Preparing a URD is one of those skills that lies between a science and an art, requiring both software technical skills and interpersonal skills.

3.3.3 BASIC OPERATIONAL REQUIREMENTS

Operational requirement is the process of linking strategic goals and objectives to tactic goals and objectives. It describes milestones, conditions for success and explains how, or what portion of, a strategic plan will be put into operation during a given operational period, in the case of, a strategic plan will be put into operation during a given operational period, in the case of commercial application, a fiscal year or another given budgetary term.

An operational plan is the basis for, and justification of an annual operating budget request. Therefore, a five-year strategic plan would typically require five operational plans funded by five operating budgets. Operational plans should establish the activities and budgets for each part of the organization for the next 1-3 years. They link the strategic plan with the activities the organization will deliver and the resources required to deliver them. An operational plan draws directly from agency and program strategic plans to describe agency and program missions and goals, program objectives, and program activities. Like a strategic plan, an operational plan addresses four questions:

• Where are we now?

• Where do we want to be?

• How do we get there?

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points:

- Mission profile or scenario: It describes about the procedures used to accomplish mission objective. It also finds out the effectiveness or efficiency of the system.
- Performance and related parameters: It points out the critical system parameters to accomplish the mission • Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

• Operational life cycle: It defines the system lifetime

SOFTWARE QUALITY ATTRIBUTES

- **Functionality**: the capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.

• Reliability: the capability of the software to maintain its level of performance when used under specified conditions. • Usability: the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.

• Efficiency: the capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.

• **Maintainability:** the capability of the software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications

. • Portability: the capability of software to be transferred from one environment to another

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

• Operational Feasibility

• Economical Feasibility

• Technical Feasibility

• Social Feasibili

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customers requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

4.2 SYSTEM DEVELOPMENT METHODOLOGY

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

4.2.1 MODEL PHASES

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

• **Requirement Analysis:** This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.

• **System Design**: Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm, data structure, software architecture etc.

• **Coding:** In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.

• **Implementation:** The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and

additional software documentation

• **Testing**: In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

• **Maintenance:** The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

REASON FOR CHOOSING WATERFALL MODEL AS DEVELOPMENT METHOD

• Clear project objectives.

• Stable project requirements.

• Progress of system is measurable.

• Strict sign-off requirements.

• Helps you to be perfect.

• Logic of software development is clearly understood.

• Production of a formal specification.

• Better resource allocation.

• Improves quality: The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.

• Less human resources required as once one phase is finished those people can start working on to the next phase.

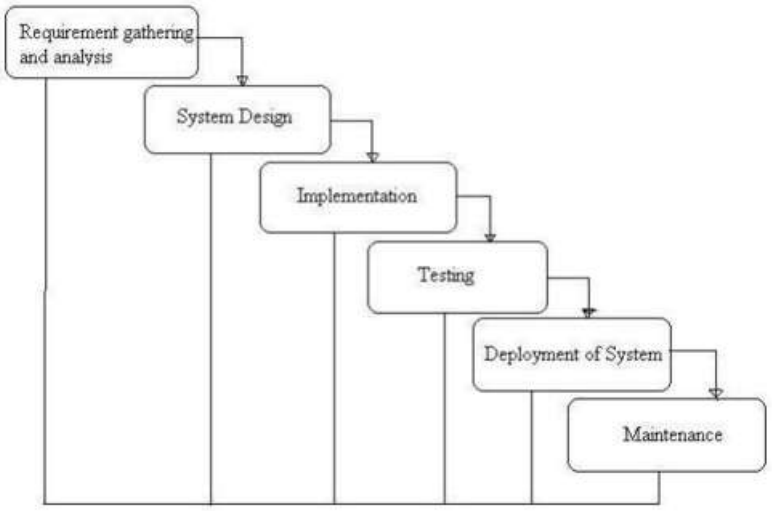

Fig: 4.1 Waterfall Model

4.4 DESIGN USING UML

Designing UML diagram specifies, how the process within the system communicates along with how the objects with in the process collaborate using both static as well as dynamic UML diagrams since in this ever-changing world of Object Oriented application development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. As a result of this challenge and the need for a universal object modeling language every one could use, the Unified Modeling Language (UML) is the Information industries version of blue print. It is a method for describing the systems architecture in detail. Easier to build or maintains system, and to ensure that the system will hold up to the requirement changes.

4.4.1 DATA FLOW DIAGRAM

A data flow diagram (DFD) is graphic representation of the ”flow” of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a contextlevel DFD first which shows the interaction between the system and outside entities. DFDs show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage. There are only four symbols: 1. Squares representing external entities, which are sources and destinations of information entering and leaving the system. 2. Rounded rectangles

representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it. 3. Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly. 4. The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing.

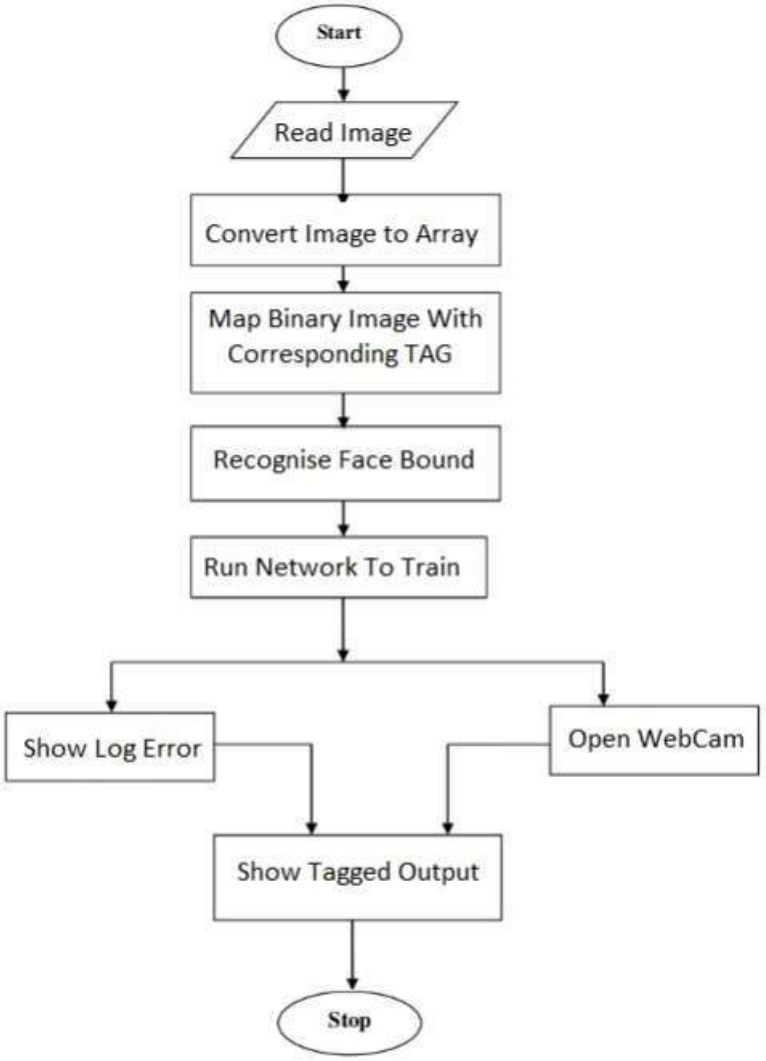

Fig: 4.2 Data flow diagram

4.4.2 USE CASE DIAGRAM

A use case defines a goal-oriented set of interactions between external entities and the system under consideration. The external entities which interact with the system are its actors. A set of use cases describe the complete functionality of the system at a particular level of detail and it can be graphically denoted by the use case diagram.

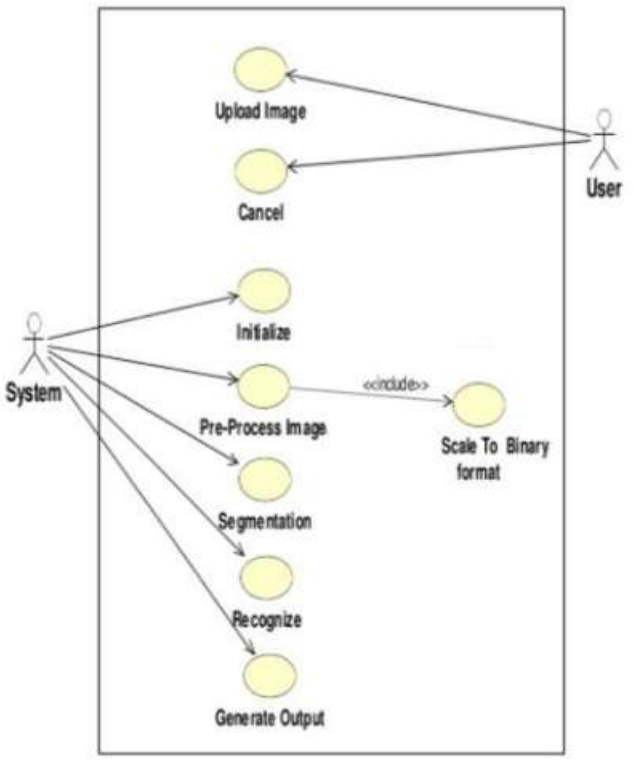

Fig 4.3 Use case Diagram

4.4.3 ACTIVITY DIAGRAM

An activity diagram shows the sequence of steps that make up a complex process. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of the activity symbol indicates a transition triggered by the completion. Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

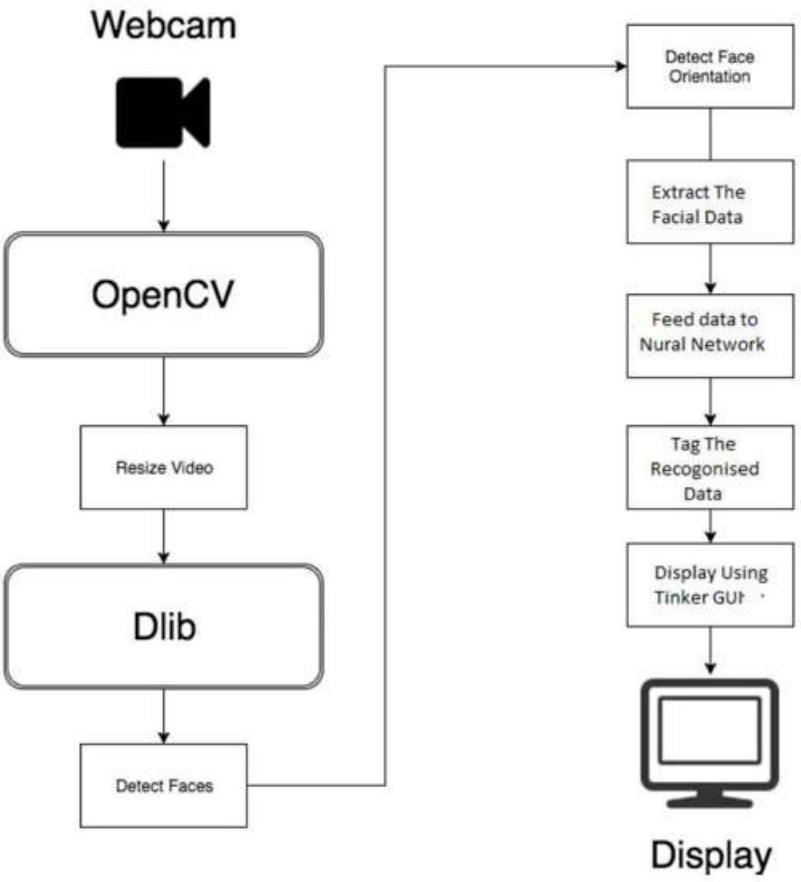

Fig 4.4 Activity Diagram

4.4.4 SEQUENCE DIAGRAM

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension represents the objects existence during the interaction.

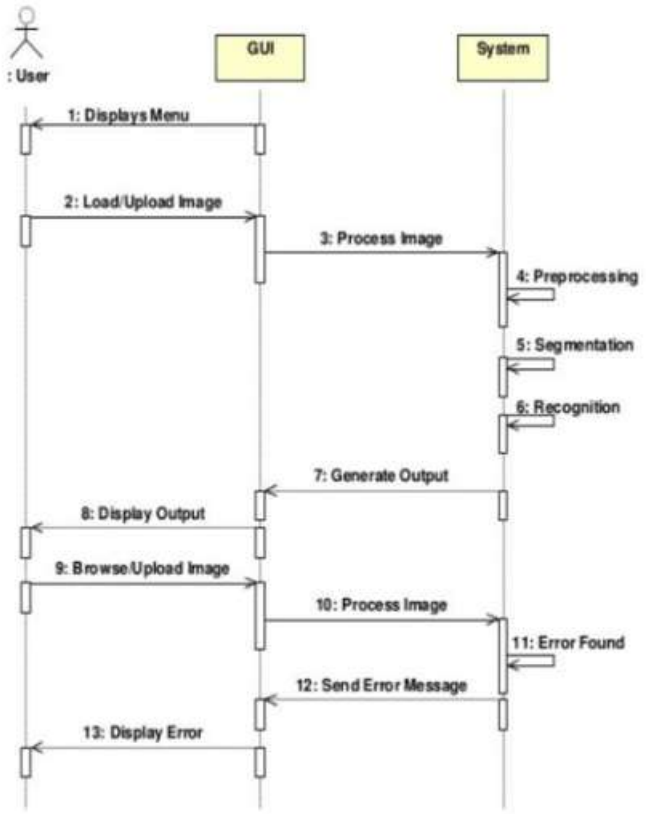

Fig 6.5: Activity Diagram

Chapter 5

IMPLEMENTATION

5.1 INTRODUCTION

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to translate the design into a best possible solution in a suitable programming language. This chapter covers the implementation aspects of the project, giving details of the programming language and development environment used. It also gives an overview of the core modules of the project with their step by step flow. The implementation stage requires the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.
- Correct decisions regarding selection of the platform.
- Appropriate selection of the language for application development.

5.2 CODING OF IMAGE TO PENCIL SKETCH

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import *
from tkinter import filedialog
from tkinter.filedialog import askopenfile
from PIL import Image, ImageTk
#from imageai.Detection import ObjectDetection

my_w = tk.Tk()
my_w.geometry("700x500") # Size of the window
#l1 = tk.Label(my_w,text='Upload Files &
display',width=30,font=my_font1)
#l1.grid(row=1,column=1,columnspan=4)
b1 = tk.Button(my_w, text='Upload Files', bg='skyblue',fg='red',
   width=20,command = lambda:upload_file())
b1.grid(row=5,column=1,columnspan=4)
b1.place(x=300,y=300)
```

```
b2 = tk.Button(my_w, text='Live Image Detection',
bg='skyblue',fg='red',
   width=20,)
b2.grid(row=5,column=1,columnspan=4)
b2.place(x=280,y=370)


def upload_file():
    f_types = [('Jpg Files', '*.jpg'),
    ('PNG Files','*.png')]   # type of files to select
    filename =
tk.filedialog.askopenfilename(multiple=True,filetypes=f_types)
    col=1 # start from column 1
    row=3 # start from row 3
    for f in filename:
        img=Image.open(f) # read the image file
        img=img.resize((200,200)) # new width & height

        img=ImageTk.PhotoImage(img)
        e1 =tk.Label(my_w)
        e1.grid(row=row,column=col)
        exit_button = tk.Button(my_w, text="Convert",
command=my_w.destroy,width=15, bg='skyblue')
        exit_button.place(x=300,y=100)
        e1.image = img # keep a reference! by attaching it to a widget
attribute
        e1['image']=img # Show Image
        e1.place(x=30, y=30)
        if(col==3): # start new line after third column
            row=row+1# start wtih next row
            col=1    # start with first column
        else:       # within the same row
            col=col+1 # increase to next column


my_w.mainloop()  # Keep the window open

plt.style.use('seaborn')

img = cv2.imread("ferrari.jpg")
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.axis("off")
plt.title("Original Image")

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray,cmap="gray")
plt.axis("off")
plt.title("GrayScale Image")
```

```
img_invert = cv2.bitwise_not(img_gray)
plt.imshow(img_invert,cmap="gray")
plt.axis("off")
plt.title("Inverted Image")


img_smoothing = cv2.GaussianBlur(img_invert, (21, 21),sigmaX=0,
sigmaY=0)

###plt.figure(figsize=(8,8))

plt.imshow(img_smoothing,cmap="gray")
plt.axis("off")
plt.title("Smoothen Image")

final = cv2.divide(img_gray, 255 - img_smoothing, scale=255)
#plt.figure(figsize=(8,8))

plt.axis("off")
plt.title("Final Sketch Image")
plt.imshow(final,cmap="gray")

#im=plt.imread("frame2.png")
#plt.imshow(im)


plt.figure(figsize=(20,20))
plt.subplot(1,5,1)
plt.imshow(img)

plt.axis("off")
plt.title("Original Image")
plt.subplot(1,5,2)
plt.imshow(img_gray,cmap="gray")

plt.axis("off")
plt.title("GrayScale Image")
plt.subplot(1,5,3)
plt.imshow(img_invert,cmap="gray")

plt.axis("off")
plt.title("Inverted Image")
plt.subplot(1,5,4)
plt.imshow(img_smoothing,cmap="gray")

plt.axis("off")
plt.title("Smoothen Image")

plt.subplot(1,5,5)
plt.imshow(final,cmap="gray")
plt.axis("off")
```

```
plt.title("Final Sketch Image")
plt.show()
```

5.3 OUTPUT

1. **Select image from a device.**

Fig 5.1

2. **Now click on a convert button.**

Fig 5.2

3. **Image Processing.**

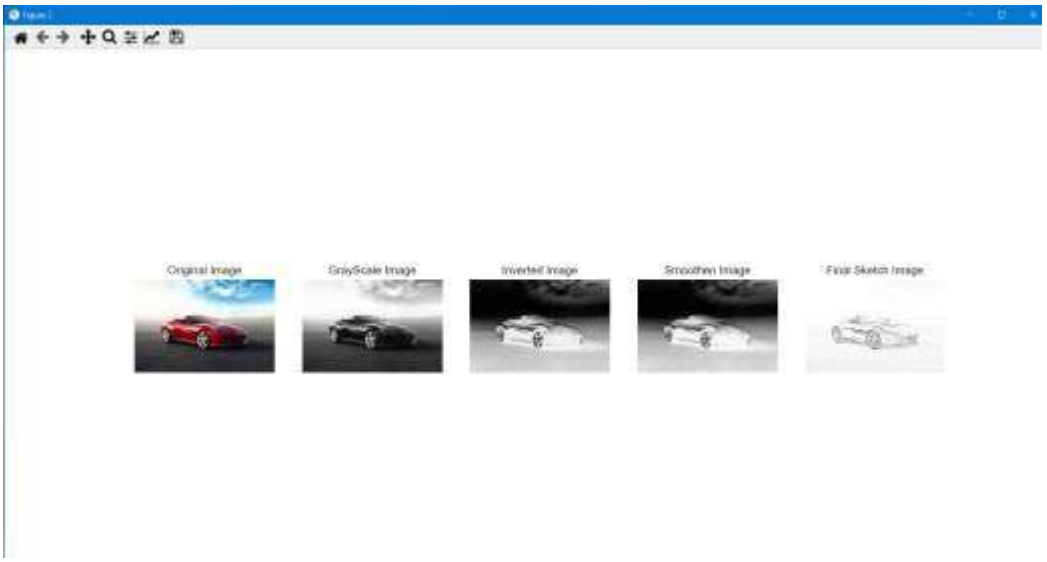

Fig 5.3

4. **Final Output.**

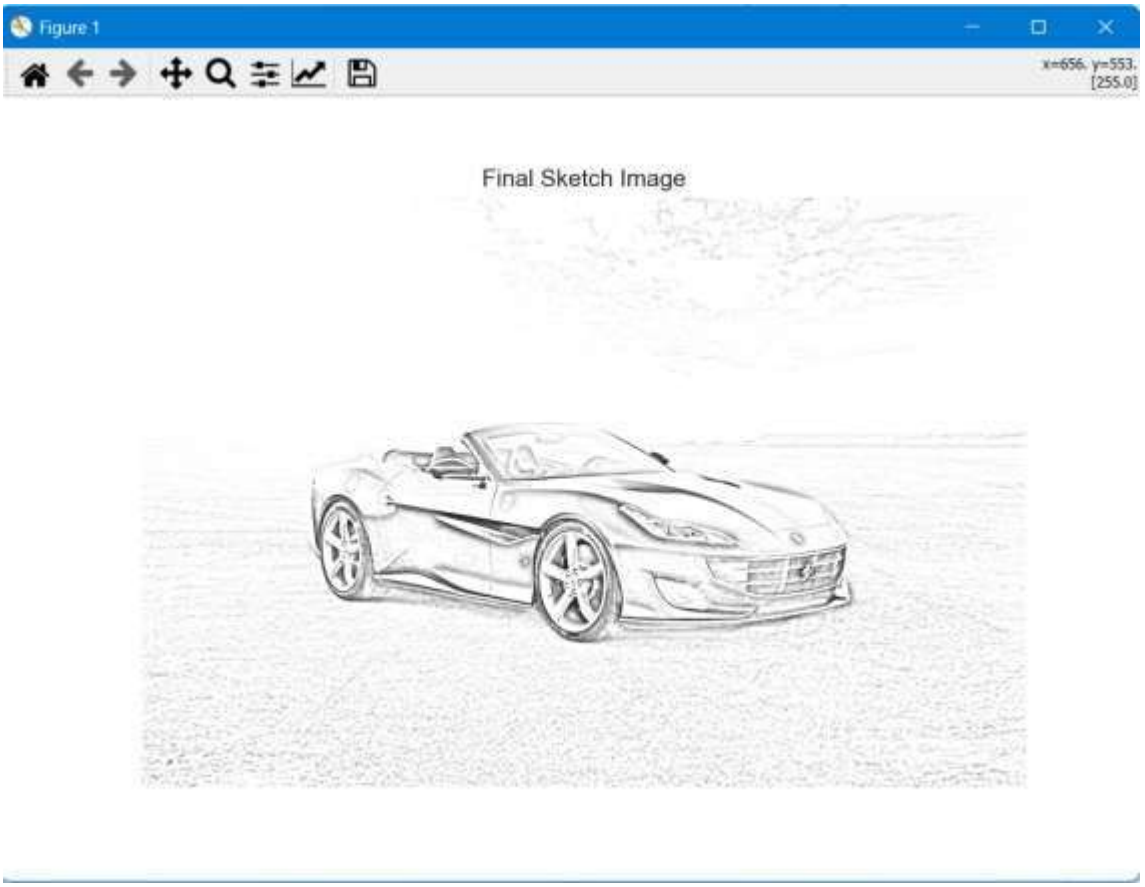

Fig 5.4

5.3 CODING OF IMAGE DETECTION

```
import pixellib
from pixellib.instance import instance_segmentation

segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
segment_image.segmentImage("path_to_image", output_image_name =
"output_image_path")

import pixellib
from pixellib.instance import instance_segmentation

segment_image = instance_segmentation()

segment_image.load_model("mask_rcnn_coco.h5")

segment_image.segmentImage("path_to_image", output_image_name =
"output_image_path")
```

Observing each line of code:

```
import pixellib
from pixellib.instance import instance_segmentation

segment_image = instance_segmentation()
```

The class for performing instance segmentation is imported and we created an instance of the class.

```
segment_image.load_model("mask_rcnn_coco.h5")
```

This is the code to load the mask rcnn model to perform instance segmentation.

```
segment_image.segmentImage("path_to_image", output_image_name = "output_image_path")
```

This is the code to perform instance segmentation on an image and it takes two parameters:

path_to_image*:* The path to the image to be predicted by the model.

output_image_name*:* The path to save the segmentation result. It will be saved in your current working directory.

OBJECT DETECTION IN VIDEO:

```
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
# initialize the video stream, allow the cammera sensor to warmup,
# and initialize the FPS counter
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
fps = FPS().start()
# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    # grab the frame dimensions and convert it to a blob
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
        0.007843, (300, 300), 127.5)
    # pass the blob through the network and obtain the detections and
    # predictions
    net.setInput(blob)
    detections = net.forward()
```

```
 # loop over the detections
    for i in np.arange(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the prediction
        confidence = detections[0, 0, i, 2]
        # filter out weak detections by ensuring the `confidence` is
        # greater than the minimum confidence
        if confidence > args["confidence"]:
            # extract the index of the class label from the
            # `detections`, then compute the (x, y)-coordinates of
            # the bounding box for the object
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            # draw the prediction on the frame
            label = "{}: {:.2f}%".format(CLASSES[idx],
                confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                COLORS[idx], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break
    # update the FPS counter
    fps.update()
 # stop the timer and display FPS information
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()

# import the necessary packages
import numpy as np
import argparse
import imutils
import time
import cv2
import os
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", required=True,
    help="path to input video")
ap.add_argument("-o", "--output", required=True,
    help="path to output video")
```

```
ap.add_argument("-y", "--yolo", required=True,
    help="base path to YOLO directory")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
    help="minimum probability to filter weak detections")
ap.add_argument("-t", "--threshold", type=float, default=0.3,
    help="threshold when applyong non-maxima suppression")
args = vars(ap.parse_args())
# load the COCO class labels our YOLO model was trained on
labelsPath = os.path.sep.join([args["yolo"], "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")
# initialize a list of colors to represent each possible class label
np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
    dtype="uint8")
# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])
configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])
# load our YOLO object detector trained on COCO dataset (80 classes)
# and determine only the *output* layer names that we need from YOLO
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# initialize the video stream, pointer to output video file, and
# frame dimensions
vs = cv2.VideoCapture(args["input"])
writer = None
(W, H) = (None, None)
# try to determine the total number of frames in the video file
try:
    prop = cv2.cv.CV_CAP_PROP_FRAME_COUNT if imutils.is_cv2() \
        else cv2.CAP_PROP_FRAME_COUNT
    total = int(vs.get(prop))
    print("[INFO] {} total frames in video".format(total))
# an error occurred while trying to determine the total
# number of frames in the video file
except:
    print("[INFO] could not determine # of frames in video")
    print("[INFO] no approx. completion time can be provided")
    total = -1
 # loop over frames from the video file stream
while True:
    # read the next frame from the file
    (grabbed, frame) = vs.read()
    # if the frame was not grabbed, then we have reached the end
    # of the stream
    if not grabbed:
        break
    # if the frame dimensions are empty, grab them
    if W is None or H is None:
```

```
        (H, W) = frame.shape[:2]

 # construct a blob from the input frame and then perform a forward
    # pass of the YOLO object detector, giving us our bounding boxes
    # and associated probabilities
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
        swapRB=True, crop=False)
    net.setInput(blob)
    start = time.time()
    layerOutputs = net.forward(ln)
    end = time.time()
    # initialize our lists of detected bounding boxes, confidences,
    # and class IDs, respectively
    boxes = []
    confidences = []
    classIDs = []
# loop over each of the layer outputs
    for output in layerOutputs:
        # loop over each of the detections
        for detection in output:
            # extract the class ID and confidence (i.e., probability)
            # of the current object detection
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]
            # filter out weak predictions by ensuring the detected
            # probability is greater than the minimum probability
            if confidence > args["confidence"]:
                # scale the bounding box coordinates back relative to
                # the size of the image, keeping in mind that YOLO
                # actually returns the center (x, y)-coordinates of
                # the bounding box followed by the boxes' width and
                # height
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) = box.astype("int")
                # use the center (x, y)-coordinates to derive the top
                # and and left corner of the bounding box
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))
                # update our list of bounding box coordinates,
                # confidences, and class IDs
                boxes.append([x, y, int(width), int(height)])
                confidences.append(float(confidence))
                classIDs.append(classID)
    # apply non-maxima suppression to suppress weak, overlapping
    # bounding boxes
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],
        args["threshold"])
    # ensure at least one detection exists
    if len(idxs) > 0:
        # loop over the indexes we are keeping
```

```
        for i in idxs.flatten():
            # extract the bounding box coordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            # draw a bounding box rectangle and label on the frame
            color = [int(c) for c in COLORS[classIDs[i]]]
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            text = "{}: {:.4f}".format(LABELS[classIDs[i]],
                confidences[i])
            cv2.putText(frame, text, (x, y - 5),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
    # check if the video writer is None
    if writer is None:
        # initialize our video writer
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 30,
            (frame.shape[1], frame.shape[0]), True)
        # some information on processing single frame
        if total > 0:
            elap = (end - start)
            print("[INFO] single frame took {:.4f} seconds".format(elap))
            print("[INFO] estimated total time to finish: {:.4f}".format(
                elap * total))
    # write the output frame to disk
    writer.write(frame)
# release the file pointers
print("[INFO] cleaning up...")
writer.release()
vs.release()
```

Make a Prediction and Interpret Result

The output of the model is, in fact, encoded candidate bounding boxes from three different grid sizes, and the boxes are defined the context of anchor boxes, carefully chosen based on an analysis of the size of objects in the MSCOCO dataset.

The script provided by experiencor provides a function called *decode_netout()* that will take each one of the NumPy arrays, one at a time, and decode the candidate bounding boxes and class predictions. Further, any bounding boxes that don't confidently describe an object (e.g. all class probabilities are below a threshold) are ignored. We will use a probability of 60% or 0.6. The function returns a list of *BoundBox* instances that define the corners of each bounding box in the

context of the input image shape and class probabilities.

```
1  # define the anchors
2  anchors = [[116,90, 156,198, 373,326], [30,61, 62,45, 59,119], [10,13, 16,30, 33,23]]
3   # define the probability threshold for detected
objects4 class_threshold = 0.6
```

```
boxes = list()
 for i in range(len(yhat)):
 # decode the output of the network
 boxes += decode_netout(yhat[i][0], anchors[i], class_threshold, input_h, input_w)
```

Next, the bounding boxes can be stretched back into the shape of the original image. This is helpful as it means that later we can plot the original image and draw the bounding boxes, hopefully detecting real objects.

The experiencor script provides the *correct_yolo_boxes()* function to perform this translation of bounding box coordinates, taking the list of bounding boxes, the original shape of our loaded photograph, and the shape of the input to the network as arguments. The coordinates of the bounding boxes are updated directly.

```
# correct the sizes of the bounding boxes for the shape of the image
correct_yolo_boxes(boxes, image_h, image_w, input_h, input_w)
```

The model has predicted a lot of candidate bounding boxes, and most of the boxes will be referring to the same objects. The list of bounding boxes can be filtered and those boxes that overlap and refer to the same object can be merged. We can define the amount of overlap as a configuration parameter, in this case, 50% or 0.5. This filtering of bounding box regions is generally referred to as non-maximal suppression and is a required post-processing step.

The experiencor script provides this via the *do_nms()* function that takes the list of bounding boxes and a threshold parameter. Rather than purging the overlapping boxes, their predicted probability for their overlapping class is cleared. This allows the boxes to remain and be used if they also detect another object type.

```
# suppress non-maximal boxes
do_nms(boxes, 0.5)
```

This will leave us with the same number of boxes, but only very few of interest. We can retrieve just those boxes that strongly predict the presence of an object: that is are more than 60% confident. This can be achieved by enumerating over all boxes and checking the class prediction values. We can then look up the corresponding class label for the box and add it to the list. Each box must be considered for each class label, just in case the same box strongly predicts more than one object.

We can develop a *get_boxes()* function that does this and takes the list of boxes, known labels, and our classification threshold as arguments and returns parallel lists of boxes, labels, and scores.

```
# get all of the results above a threshold
def get_boxes(boxes, labels, thresh):
 v_boxes, v_labels, v_scores = list(), list(), list()
 # enumerate all boxes
 for box in boxes:
 # enumerate all possible labels
 for i in range(len(labels)):
 # check if the threshold for this label is high enough
 if box.classes[i] > thresh:
```

```
    v_boxes.append(box)
    v_labels.append(labels[i])
    v_scores.append(box.classes[i]*100)
    # don't break, many labels may trigger for one box
    return v_boxes, v_labels, v_scores
```

We can call this function with our list of boxes.

We also need a list of strings containing the class labels known to the model in the correct order used during training, specifically those class labels from the MSCOCO dataset. Thankfully, this is provided in the experiencor script.

```
# define the labels
labels = ["person", "bicycle", "car", "motorbike", "aeroplane", "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", "sofa", "pottedplant", "bed", "diningtable", "toilet", "tvmonitor", "laptop", "mouse",
    "remote", "keyboard", "cell phone", "microwave", "oven", "toaster", "sink", "refrigerator",
    "book", "clock", "vase", "scissors", "teddy bear", "hair drier", "toothbrush"]
# get the details of the detected objects
v_boxes, v_labels, v_scores = get_boxes(boxes, labels, class_threshold)
```

Now that we have those few boxes of strongly predicted objects, we can summarize them.

```
# summarize what we found
for i in range(len(v_boxes)):
    print(v_labels[i], v_scores[i])
```

We can also plot our original photograph and draw the bounding box around each detected object. This can be achieved by retrieving the coordinates from each bounding box and creating a Rectangle object.

```
box = v_boxes[i]
# get coordinates
y1, x1, y2, x2 = box.ymin, box.xmin, box.ymax, box.xmax
# calculate width and height of the box
width, height = x2 - x1, y2 - y1
# create the shape
rect = Rectangle((x1, y1), width, height, fill=False, color='white')
# draw the box
ax.add_patch(rect)
```

We can also draw a string with the class label and confidence.

```
# draw text and score in top left corner
label = "%s (%.3f)" % (v_labels[i], v_scores[i])
pyplot.text(x1, y1, label, color='white')
```

The *draw_boxes()* function below implements this, taking the filename of the original photograph and the parallel lists of bounding boxes, labels and scores, and creates a plot showing all detected objects.

```
# draw all results
def draw_boxes(filename, v_boxes, v_labels, v_scores):
 # load the image
 data = pyplot.imread(filename)
 # plot the image
 pyplot.imshow(data)
 # get the context for drawing boxes
 ax = pyplot.gca()
 # plot each box
 for i in range(len(v_boxes)):
 box = v_boxes[i]
 # get coordinates
y1, x1, y2, x2 = box.ymin, box.xmin, box.ymax, box.xmax
 # calculate width and height of the box
width, height = x2 - x1, y2 - y1
 # create the shape
rect = Rectangle((x1, y1), width, height, fill=False, color='white')
 # draw the box
 ax.add_patch(rect)
 # draw text and score in top left corner
 label = "%s (%.3f)" % (v_labels[i], v_scores[i])
 pyplot.text(x1, y1, label, color='white')
# show the plot
 pyplot.show()
```

We can then call this function to plot our final result.

```
# draw what we found
draw_boxes(photo_filename, v_boxes, v_labels, v_scores)
```

We now have all of the elements required to make a prediction using the YOLOv3 model, interpret the results, and plot them for review.

OUTPUT

1. **Select Input image.**

Fig 5.5

2. **Processing Image.**

3. **Final Image.**

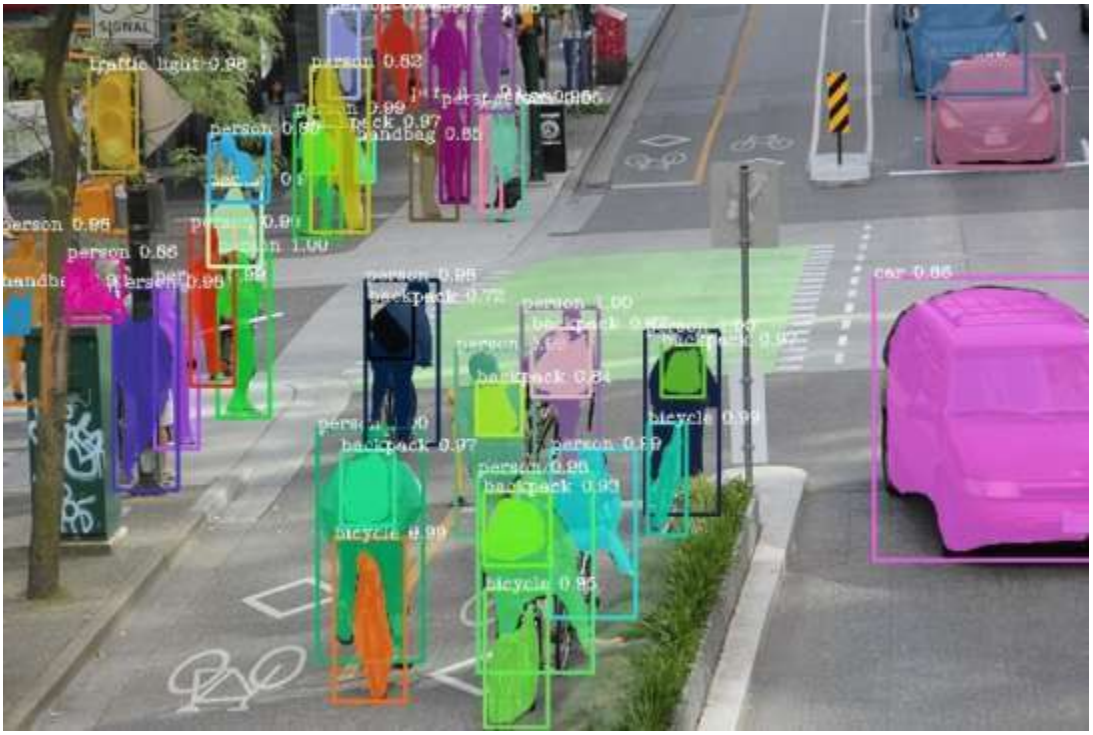

Fig 5.7

Now Start implementation with ImageAI

Now let's see how to actually use the ImageAI library. I will explain step by step how you can build your first project based on object detection model with ImageAI.

Step 1:-

Create an object_detect.py or object_detect.ipynb file that your preferred text editor for writing Python code.

Step 2:-

Import *object* detection library

```
from imageai.Detection import ObjectDetection
```

Step 3:-

Now that you have imported imageAI library and the ObjectDetection class, the next thing is to create an instance of the class object detection.

detector = ObjectDetection()

NOTE:- if error while running the ImageAI library then update the keras>= 1.4 and TensorFlow = 2.2.0.

Step 4:-

Let's set the path of input file, output file, and model file.

#path of each file

models_path = "/content/yolo-tiny.h5"

input_path = "/content/input_test.jpg"

output_path = "/content/output_test.jpg

#path of each file

models_path = "/content/yolo-tiny.h5"

input_path = "/content/input_test.jpg"output_path = "/content/output_test.jpg

Step 6:-

After instantiating the ObjectDetection class we can now call various functions from the class. The class contains the following functions to call pre-trained models: setModelTypeAsRetinaNet(), setModelTypeAsYOLOv3(), and setModelTypeAsTinyYOLOv3().

For this tutorial, I will be using the pre-trained TinyYOLOv3 model, and hence we will use the setModelTypeAsTinyYOLOv3() function to load our model.

detector.setModelTypeAsTinyYOLOv3()

Next, we are going to call the function setModelPath(). This function accepts a string which contains the path to the pre-trained model:

detector.setModelPath(models_path)

This step calls the function loadModel() from the detector instance. It loads the model from the path specified above using the setModelPath() class method.

detector.loadModel()

Step 7:-

To detect objects in the image, we need to call the detectObjectsFromImage function using the detector object that we created in the previous section.

This function requires two arguments: input_image and output_image_path. input_image is the path where the image we are detecting is located, while the output_image_path parameter is the path to store the image with detected objects. This function returns a dictionary that contains the names and percentage probabilities of all the objects detected in the image.

```
detection = detector.detectObjectsFromImage(input_image=input_path,
output_image_path=output_path)
```

Step 8:-

The dictionary items can be accessed by traversing through each item in the dictionary.

```
for eachItem in detection:
  print(eachItem[“name”] , ” : “, eachItem[“percentage_probability”])
```

Chapter 6

MODULES DISCRIPTION

IMAGE TO PENCIL SKETCH

1. *Select File:* This is a module in which the application will select the file from the user data. The user select their own choice of image that it want to convert into a beautiful sketch.

2. *Convert File:* In this module the application can be able to convert a user selected image into a beautiful sketch.

3. *Image Processing Algorithm:* This module helps the image to for processing to make sketch of image.

IMAGE TO PENCIL SKETCH

1. *Camera Module:* In this Module, Your device camera can recognized the things near you.
2. *Image Processing Algorithm:* This module helps the image to for processing to make sketch of image.
3. *Output Generation*: This module helps to detect the live images and process these image, after processing it gives a live output on a screen.

FUTURE SCOPE

The making of sketch of the any images with a traditional method is complex. To make a sketch with a traditional method, an artist required the most important things like patience, time, and keep an eye for detail and composition.

The future of this project will help the peoples, artist, etc. to make easily the beautiful and attractive sketch of any image. They can simple make a sketch by selecting the image in a user files and get a output as a sketch. The technology we used in this project help the peoples to easily interact with the application. It also help the people to not depend on the artist to make their won sketch. They can easily perform the task by interacting with a application.

The object recognition system can be applied in the area of surveillance system, face recognition,fault detection, character recognition etc. The objective of this thesis is to develop an object recognition system to recognize the 2D and 3D objects in the image. The performance of the object recognition system depends on the features used and the classifier employed for recognition. This research work attempts to propose a novel feature extraction method for extracting global features and and obtaining local features from the region of interest. Also the research work attempts to hybrid the traditional classifiers to recognize the object.

- Design and simulation of complex video sequence and test them using same tracking algorithm. In the potential scenario, occlusion is used for an object with the same color for the moving objects or else using bigger occlusion with longer occlusion time. Increasing the number of the object help to identify the efficiency and functionality of the tracking algorithm.
- Weight parameters are needed to be added for individual intensity levels of each pixel. In an image, if an intensity value is assigned as foreground based on the current frame then it has less probability that foreground also has similar pixel coordinate so that BG weightage for the pixel is set to the minimum than the initial value. Through adding weightage lower than the initial value provides the advantage of removing the old pixel value with least probability rather than the

evolved scene.

- Need to focus towards enhancing the variance data of each channel based on the Mahalanobis distance calculation. By this, can able to adopt a change in the rapid scene through Euclidean distance algorithm.

CONCLUSION

I would like to conclude that, it is emmensive learning experience while preparing the project. This is achieved through an easy to use graphical interface python libraries like Tkinter, matplotlib, etc. User can simply select the image from their files. After that it can click on the convert button. The image is converted into a beautiful sketch.

The main motive behind the project is to improve human computer interaction by developing an approach to help people by converting their different images into a beautiful sketch. A good user interface is provided to the user to upload a sketch input image and receive related output images. By using this thesis and based on experimental results we are able to detect object more precisely and identify the objects individually with exact location of an object in the picture in x,y axis. This paper also provide experimental results on different methods for object detection and identification and compares each method for their efficiencies.

In this project, review on different object detection, tracking, recognition techniques, feature descriptors and segmentation method which is based on the video frame and various tracking technologies. This approach used towards increase the object detection with new ideas. Furthermore, tracking the object from the video frames with theoretical explanation is provided in bibliography content. The bibliography content is the most significant contribution of research since it will lead to a new area of research. We have identified and discussed the limitation/future scope of various methods. Also, we have noted some methods which give accuracy but have high computational complexity. Specifically, the statistical methods, background subtraction, temporal differencing with the optical flow was discussed. However, these technique needs to concentrate towards handling sudden illumination changes, darker shadows and object occlusions

REFERENCES

1. [Lu et al. 2012] C. Lu, L. Xu and J. Jia, Combining Sketch and Tone for Pencil Drawing Production. International Symposium on Non-Photorealistic Animation and Rendering (NPAR), 2012. [McCloud 1993] Scott McCloud, Understanding Comics. Harper Collins Publishers, New York, 1993.
2. Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Mach. Intell. 26,1475–1490.
3. doi:10.1109/TPAMI.2004.108
4. Alexe, B., Deselaers, T., and Ferrari, V. (2010). "What is an object?," in ComputerVision and Pattern Recognition (CVPR), 2010 IEEE Conference on (San Francisco,CA: IEEE), 73–80.doi:10.1109/CVPR.2010.5540226
5. Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. Int. J.Comput. Vis. 1, 333–356. doi:10.1007/BF00133571
6. Andreopoulos, A., and Tsotsos, J. K. (2013). 50 years of object recognition: direc-tions forward. Comput. Vis. Image Underst. 117, 827–891. doi:10.1016/j.cviu.2013.04.005
7. Azizpour, H., and Laptev, I. (2012). "Object detection using strongly-superviseddeformable part models," in Computer Vision-ECCV 2012 (Florence: Springer),836–849.
8. Azzopardi, G., and Petkov, N. (2013). Trainable cosfire filters for keypoint detectionand pattern recognition. IEEE Trans. Pattern Anal. Mach. Intell. 35, 490–503.doi:10.1109/TPAMI.2012.106
9. Azzopardi, G., and Petkov, N. (2014). Ventral-stream-like shape representation:from pixel intensity values to trainable object-selective cosfire models. Front.Comput. Neurosci. 8:80.doi:10.3389/fncom.2014.00080
10. Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). "Fast classification using sparsedecision dags,"in Proceedings of the 29th International Conference on MachineLearning (ICML-12), ICML '12, eds Langford and J. Pineau (New York,

NY:Omnipress), 951–958.

11. Bengio, Y. (2012). “Deep learning of representations for unsupervised and transferlearning,” in ICML Unsupervised and Transfer Learning, Volume 27 of JMLRProceedings, eds I. Guyon, G. Drom V. Lemaire, G. W. Taylor, and D. L. Silver(Bellevue: JMLR.Org), 17–36.
12. Bourdev, L. D., Maji, S., Brox, T., and Malik, J. (2010). “Detecting peopleusing mutually consistent poselet activations,” in Computer Vision – ECCV2010 – 11th European Conference on Computer Vision, Heraklion, Crete, Greece,September 5-11, 2010, Proceedings, Part VI, Volume 6316 of Lecture Notes inComputer Science, eds K. Daniilidis, P. Maragos, and N. Paragios(Heraklion:Springer), 168–181.
13. Bourdev, L. D., and Malik, J. (2009). “Poselets: body part detectors trained using 3dhuman pose annotations,” in IEEE 12th International Conference on ComputerVision, ICCV 2009, Kyoto, Japan,September 27 – October 4, 2009 (Kyoto: IEEE),1365–1372.
14. Abdul Muhsin M, Farah F. Alkhalid, Bashra Kadhim Oleiwi, “Online Blind Assistive System using Object Recognition”, International Research Journal of Innovations in Engineering and Technology (IRJIET) ,Volume 3, Issue 12, pp 47-51, December – 2019.
15. Aishwarya Sarkale, Kaiwant Shah, Anandji Chaudhary, Tatwadarshi P. N., “A Literature Survey: Neural Networks for Object Detection”, VIVA-Tech International Journal for Research and Innovation Volume 1, Issue 1 (2018) ISSN(Online): 2581-7280 Article No. 9.
16. Bhumika Gupta, Ashish Chaube, Ashish Negi, Umang Goel, “Study on Object Detection using Open CV Python”, International Journal of Computer Applications Foundation of Computer Science (FCS), NY, USA, Volume 162, Number 8, 2017.
17. Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam, “Real-Time Object Detection with Yolo”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-3S, February 2019.
18. Karanbir Chahal1 and Kuntal Dey, “A Survey of Modern Object Detection Literature using Deep Learning”, International Research Journal of Engineering and Technology (IRJET), Volume 8, Issue 9, 2018.
19. Kartik Umesh Sharma and Nileshsingh V. Thakur, “A Review and an Approach for Object Detection in Images”, International Journal of Computational Vision and Robotics, Volume 7, Number 1/2, 2017.

20. Ojha, S. & Sakhare, S., 2015. Image processing techniques for object tracking in video surveillance- A survey, in: 2015 International Conference on Pervasive Computing (ICPC). IEEE, pp. 1–6. doi:10.1109/PERVASIVE.2015.7087180
21. Poschmann, P., Huber, P., Rätsch, M., Kittler, J. & Böhme, H.-J., 2014. Fusion of Tracking Techniques to Enhance Adaptive Real-time Tracking of Arbitrary Objects. Procedia Comput. Sci. 39, 162–165. doi:http://dx.doi.org/10.1016/j.procs.2014.11.025
22. Ramya, P. & Rajeswari, R., 2016. A Modified Frame Difference Method Using Correlation Coefficient for Background Subtraction. Procedia Comput. Sci. 93, 478–485. doi:10.1016/j.procs.2016.07.236
23. Risha, K.P. & Kumar, A.C., 2016. Novel Method of Detecting Moving Object in Video. Procedia Technol. 24, 1055–1060. doi:10.1016/j.protcy.2016.05.235
24. Sarkar, R., Bakshi, S. & Sa, P.K., 2012. A Real-time Model for Multiple Human Faces Tracking from Low-resolution Surveillance Videos. Procedia Technol. 6, 1004–1010. doi:http://dx.doi.org/10.1016/j.protcy.2012.10.122
25. Shen, H., Li, S., Zhu, C., Chang, H. & Zhang, J., 2013. Moving object detection in aerial video based on spatiotemporal saliency. Chinese J. Aeronaut. 26, 1211–1217. doi:10.1016/j.cja.2013.07.038
26. Soundrapandiyan, R. & Mouli, P.V.S.S.R.C., 2015. Adaptive Pedestrian Detection in Infrared Images Using Background Subtraction and Local Thresholding. Procedia Comput. Sci. 58, 706–713. doi:10.1016/j.procs.2015.08.091
27. Susar, R. & Dongare, M., 2015. Moving Object Detection, a Succinct Review. Int. J. Adv. Res. Comput. Commun. Eng. 4, 334–336. doi:10.17148/IJARCCE.2015.41277
28. Tang, D. & Zhang, Y.J., 2011. Combining mean-shift and particle filter for object tracking, in: Proceedings - 6th International Conference on Image and Graphics, ICIG 2011. IEEE, pp. 771–776. doi:10.1109/ICIG.2011.118
29. Tao Zhang, Zaiwen Liu, Xiaofeng Lian & Xiaoyi Wang, 2010. Study on moving-objects detection technique in video surveillance system, in: 2010 Chinese Control and Decision Conference. IEEE, pp. 2375–2380. doi:10.1109/CCDC.2010.5498797

Internal References

1. *For Python*, https://www.w3schools.com/python/
2. *For Tkinter*, https://www.geeksforgeeks.org/python-gui-tkinter/
3. *For Matplotlib, https://matplotlib.org/cheatsheets/*