

**A PROJECT REPORT ON**

**“CUSTOM OBJECT DETECTION USING python AND open cv”**

**Submitted to**

**INTEL Unnati Industrial Training**

**Program 2024**

**Submitted By:**

**STUDENT NAME: PRATIKSHA B K**

**Project Mentor**:

**DR. SHIVAKUMAR KAGI**

# **INTRODUCTION**

Computer vision is a part of Artificial Intelligence (AI), that enables computers to gain meaningful insights from the data provided to it (the data can be in the form of images, videos, or any other visual input) and make relevant decisions based on the knowledge it acquired from the given data. Computer vision provides the computer with the ability to see the world around as humans do. It enables them to possess the intelligence to distinguish physical features and recognize objects.

Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization is refer to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term “object recognition”, they often mean “object detection”. It may be challenging for beginners to distinguish between different related computer vision tasks

So, we can distinguish between these three computer vision tasks with this example: Image Classification: This is done by Predict the type or class of an object in an image. Input: An image which consists of a single object, such as a photograph. Output: A class label (e.g. one or more integers that are mapped to class labels). Object Localization: This is done through, Locate the presence of objects in an image and indicate their location with a bounding box.

Input: An image which consists of one or more objects, such as a photograph. Output: One or more bounding boxes (e.g. defined by a point, width, and height). Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

Input: An image which consists of one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

For example, image classification is simply straight forward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just as equally referred to as object recognition.

**DATASET DESCRIPTION**

* Object detection is an important task, yet challenging vision task. It is a critical part of many applications such as image search, image auto-annotation and scene understanding, object tracking. Moving object tracking of video image sequences was one of the most important subjects in computer vision. It had already been applied in many computer vision fields, such as smart video surveillance, artificial intelligence, military guidance, safety detection and robot navigation, medical and biological application. In recent years, a number of success full single -object tracking system appeared, but in the presence of several objects, object detection becomes difficult and when objects are fully or partially occluded, they are obtruded from the human vision which further increases the problem of detection. Decreasing illumination and acquisition angle. The proposed MLP based object tracking system is made robust by an optimum selection of unique features and also by implementing the Ada boost strong classification method Upgrade pip with mentioned command below.
* Upgrade pip with mentioned command below.

**pip install –upgrade pip**

* Install libraries with mentioned command below.

**pip install cv2**

**pip install numpy**

**pip install keras**

**pip install tensorflow**

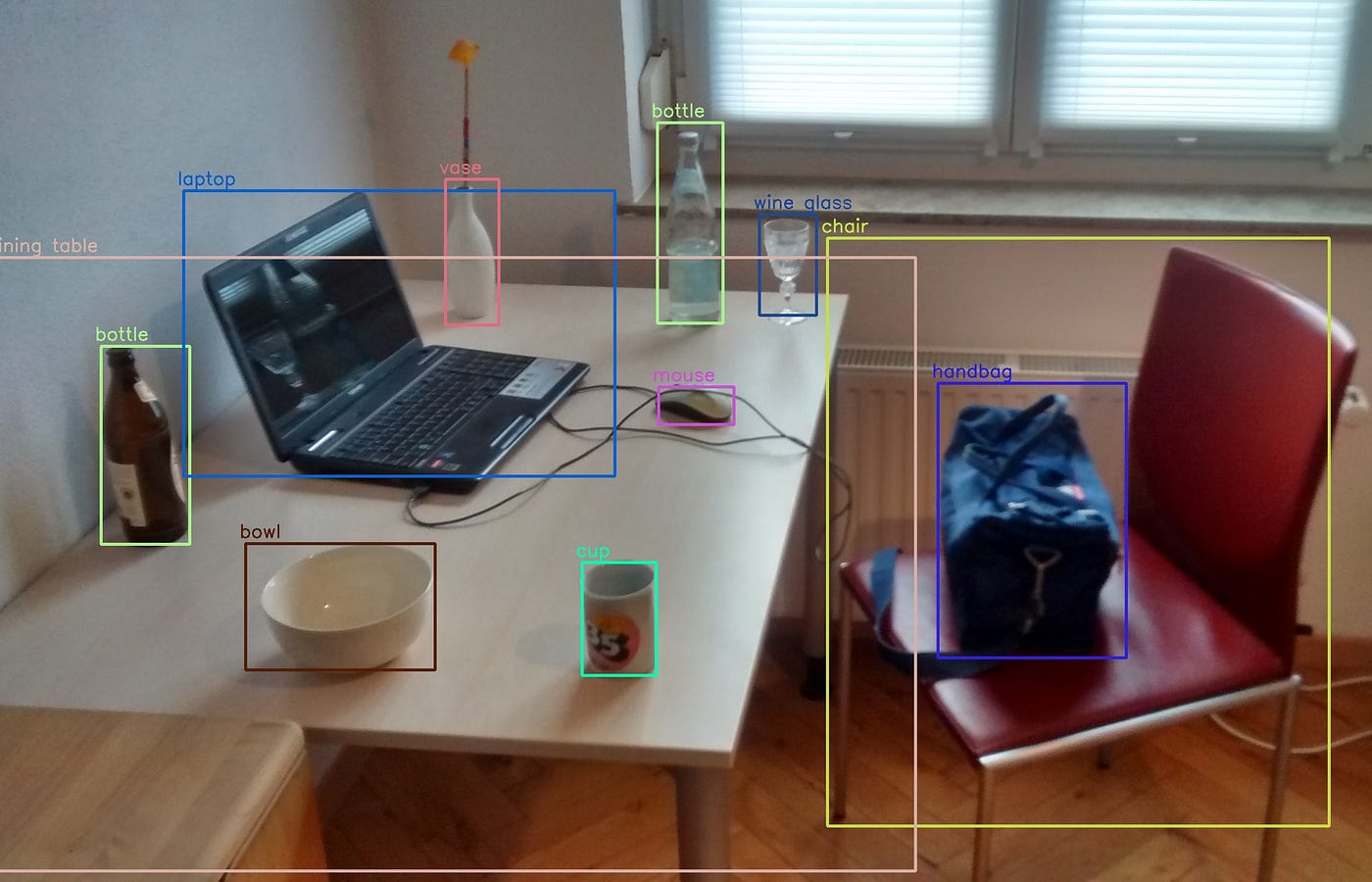
**pip install opencv python**

**METHODOLOGY**

In this project we present our views on the state-of-the-art and future technology milestones. We have put forward an interesting way for the analysis of custom object detection using python and how it makes life easier and smart in the coming decade.

Initially we have used python as a tool for running the project and then uploaded a video and images of movement of vehicles. Then we run the developed code in the VS Code and observe the output. So as mentioned earlier when we run the code in VS Code and upload the videos and images in it.

**RESULTS AND DISCUSSION**

****

**CONCLUSION**

In this work, we have discussed the concept of object detection and tracking in a video using OpenCV in Python, using various methods such as frame differencing, colours paces, background separation, optical flows, and Haar cascades classifiers. We have also discussed in detail, a famous edge detection algorithm – Canny’s Edge Detector. • We used the rich library set of OpenCV for a robust face detection from a sample video. For training the model with the feature set of a face, we used the “Haar frontal face” XML file. • We later extended our model to detect eyes and nose in the same input video. We used “haar\_eyes” and “haar\_mcs\_nose” XML files for this purpose. • We also implemented all concepts like frame differencing, colorspaces, background separation, optical flows using OpenCV in Python. • We implemented Canny’s Edge Detection algorithm also in OpenCV on a sample video and detected all the edges in the video. • All our models could successfully detect all faces, eyes, and noses in the input image with 100% detection accuracy and in real-time detection speed.

**CODE**

from keras.models import load\_model

import cv2

import numpy as np

np.set\_printoptions(suppress=True)

model = load\_model("keras\_Model.h5", compile=False)

class\_names = open("labels.txt", "r").readlines()

camera = cv2.VideoCapture(0)

while True:

ret, image = camera.read()

image = cv2.resize(image, (224, 224), interpolation=cv2.INTER\_AREA)

cv2.imshow("Webcam Image", image)

image = np.asarray(image, dtype=np.float32).reshape(1, 224, 224, 3)

image = (image / 127.5) - 1

prediction = model.predict(image)

index = np.argmax(prediction)

class\_name = class\_names[index]

confidence\_score = prediction[0][index]

print("Class:", class\_name[2:], end="")

print("Confidence Score:", str(np.round(confidence\_score \* 100))[:-2], "%")

keyboard\_input = cv2.waitKey(1)

if keyboard\_input == 27:

break

camera.release()

cv2.destroyAllWindows()

**SOLUTION FEATURES**

Data Processing:

