**International Journal of Engineering Sciences & Emerging Technologies, Oct. 2023.**
**ISSN: 22316604 Volume 11, Issue 2, pp: 223-231 ©IJESET**

# RECOGNITION AND DETECTION OF OBJECTS IN VIDEO USING OPENCV

Neha Gupta, Prakhar Kapoor, Sejal Thakur

Department of Computer Science and Engineering, AKTU University, Moradabad, India
discoverneha@gmail.com
prakharvkapoor@gmail.com
Sejal12709@gmail.com

## *ABSTRACT*

*Object detection is the process of identifying, locating, and detecting different objects. It can be done in a video or in an image. A large number of tools and technologies are used in detecting different objects. Tools like OpenCV, TensorFlow, YOLO[1] , SSD, etc. It displays the number of different objects present in the image or video. We use python as the programming language, as it has a large library that supports all the operations required for object detection. Here we have used coco dataset which consist of 80, distinct objects like person, chair, car, mobile, etc. it is a labelled dataset.*

***KEYWORDS -*** *Object detection, OpenCV, Frozen Inference Graph, Threshold Value, confidence Value*

## 1. INTRODUCTION

Object detection is the process of finding and recognizing real-world object instances such as car, bike, TV, etc. out of an image or video. An object detection technique lets you understand the details of an image or a video, as it allows you to localize and analyze multiple objects within the image/video. The object detection in video is the same as that of an image. The only sole difference between both is that we must split up the video frame by frame, and then the object detection model is applied on each frame in a chronological manner. Neural networks are used for this purpose as complex calculations and other complicated operations are performed on different segments of the image. One of the most widely used libraries for this purpose is OpenCV. This is based on Convolutional Neural Network[2] (CNN) which is an advanced type of neural network. We also use mobile net SSD algorithm that is based on the SSD algorithm. This is used for computing the output bounding box and the object class from the input image.

Some important terms are:
**OBJECT DETECTION**: it is a computer vision technique for locating instances of objects in images or video.

**OpenCV**: it is an open-source computer vision and machine learning library, built to provide a common infrastructure for computer vision application.

**OBJECT CLASSIFICATION**: it is the categorization of the object based on previously defined class or types. It does not deal with locating them.

**COCO DATASET**: it stands for common objects in context dataset. It is an image recognition dataset that has 80 object categories.

**OBJECT SEGMENTATION**: it is the technique to provide the exact outline of the object within an image or video.

DOI: 10.5281/zenodo.10935287

## 2. PREREQUISITES AND REQUIREMENTS

### 2.1. Hardware and Standard Platform

The object detection system has been on the Mac machine with the following characteristics*:*

- Processor: 1.8GHz Dual-Core Intel Core i5
- Memory: 8 GB
- GPU: intel HD Graphics 6000 1536 MB

### 2.2. SOFTWARE

The following software, Frameworks and Libraries are used for object detection:

- Programming Language: PYTHON
- Library: OpenCV
- Algorithm: Mobile net SSD
- Dataset: COCO Dataset
- IDE : Jupyter Notebook

## 3. IMPORTANT TERMS AND CONCEPTS

Object detection is a vast field of computer vision. Many techniques and complex operations are performed for object detection; hence we need to understand a few basic terminologies.

- **Edge Matching**: in this, the edges of the objects are found using edge detecting techniques. The edges are detected by observing changes in light and colour. It is to be noted that a number of overlapping edges are to be counted.
- **Divide and Conquer Search**: it creates a set of all the positions, the best position in the cell is occupied by the lower bound and as the cell becomes small it will end the process.
- **Grayscale Matching**: it is the method where a change in colour can be recognized and compared and be measured on a relative scale.
- **Edge Detection**: it is used to detect the edge of the image with the help of image segmentation and image extraction.
- **Gradient Matching**: it is the methodology or tool to match the various image gradient vector. In a comparison of images to making the image robust for more changes.
- **Image Gradient Vector**: it is defined as metric for every individual pixel, contain the pixel colour change in both x-axis and y-axis.

## 4. TOOLS USED

Various programming libraries, frameworks and other additional tools were used for object detection. These are as follows:

### 4.1. OpenCV

OpenCV[3] is a python library that allows to perform image processing and computer vision tasks.it was developed by intel in 1999. OpenCV offers a comprehensive set of tools and algorithm that empowers developers to analyze, manipulate, and understand digital images and videos. Its rich features set includes image and video input and output, image filtering, feature detection, object recognition, camera calibration, and much more. These capabilities allow for real-time processing, object tracking and even face detection. One of the notable advantages of OpenCV is its compatibility with multiple programming languages, including C++. Python, and java, making it accessible to a wide range of developers. This flexibility facilitates the integration of OpenCV into existing software projects and encourages collaborative development. The tasks done by OpenCV are image filtering, feature detection, object detection, object tracking, etc. There are various modules of the OpenCV like core, imgproc, videoio, objdetect, calib3d,dnn,features2d,etc. CV2 is the name that OpenCV developer chose when they created the binding generators. There are various advantages of OpenCV like cross-platform

DOI: 10.5281/zenodo.10935287

compatibility, large community support, integration with other libraries and frameworks, open source and free, high performance.

### 4.2. COCO DATASET

For this project we have used coco dataset. Coco dataset is called common objects in context dataset. It is a widely used benchmark in computer vision domain for the task of object detection, segmentation and captioning task. It consists of a vast collection of 300,000 images with over 80 different object classes covering a large range of everyday objects. Furthermore, it provides pixel-level annotations for object instances, including bounding boxes and segmentation masks. Coco is known for its high-level annotation and has become a standard reference for evaluating and benchmarking the object detection algorithm. It serves as a valuable resource for training and evaluating models aimed at understanding objects in complex real-world contexts. We can download coco dataset from various sites like RoboFlow, git, etc. One of the remarkable things about this dataset is that it is free and is openly available for use to train and build the object detection model. Person, car, motorbike, mobile, scissors, bottle, etc. are the few categories of the coco dataset is as follows in fig.1:


**Figure 1:** COCO dataset glimpse

### 4.3. Mobilenet SSD Selecting

Mobile net SSD[4] is an object detection model that computes output bounding box and object class from the input image. It is a framework that combines the mobile net architecture for effective feature selection with the single shot multibox detector (SSD) for accurate and precise object detection. It is developed by Google for resource constrained devices like smartphones and embedded systems. Likewise, it replaces the traditional convolutional layers with the depth wise convolutions followed by pointwise convolutions, this reduces the complexity and number of parameters. This algorithm enables real time object detection by performing the detection at multiple scales and ratio in a single forwarded pass. It uses a set of default anchor boxes to predict the location of the object and the class to which it belongs to, and then the prediction is passed to a series of convolutional layers for refinement and get better accuracy. This framework is a good balance between the accuracy and speed, making it a perfect choice for real time object detection applications. Its lightweight and effective architecture allows it to run smoothly and efficiently. Generally, this framework is highly effective when combined with the coco dataset. It is to be noted that mobile net SSD is different from SSD, as SSD only provides localization and mobile net SSD provides classification. It can reach up to 12 fps with mAP of .05 value of 86.8%

DOI: 10.5281/zenodo.10935287

### 4.4. FROZEN INFERENCE GRAPH

Frozen graphs are commonly used for inference in the TensorFlow[5] and are the steppingstones for inference for other frameworks. It defines the combination of the model graph structure with kept values of the required variables. It is a trained deep learning model that has been optimized and converted into suitable format for inference. Once the training is complete, the model is usually saved in a format that include both the architecture of the model and the weights. The model is obtained by freezing the trained model's parameters. Freezing the parameters means that we have fixed the weights, and they cannot be further updated during the inference. This process eliminates the need for backpropagation and reduces the memory footprint of the model. With OpenCV, the frozen graph can be used for object detection. OpenCV has the ability or functionality to read the frozen model and perform inferences over the images or videos. This frozen graph is generally used with pre-trained models. Additionally, it allows state – of – the – art object detection models into OpenCV based applications, enabling efficient and accurate object detection functionality and capabilities.

### 4.5. MATPLOTLIB

Matplotlib is a library of the python. It is use for plotting various different kinds of charts and graphs that are used for the pictorial representation of the data. Matplotlib is used in object detection as to print out the image or video on which the object detection is to be performed. It offers a simple and intuitive API for creating line plots, scatter plots, histograms, pie charts, etc. It also allows customization of the plot aesthetics including the colour, line, marker, labels, titles, etc. it supports coordinate system, axis scaling and subplots. It is generally used for data visualization purpose due to its extensive functionality and integration with other python libraries.

## 5. METHODOLOGY

### 5.1. Importing Necessary Libraries

First and foremost, step is to import all the libraries that are essential for your project. Libraries are the simply a collection of codes or modules of codes that we can use in a program for specific operations. We use libraries so that we don't need to write the code again in our program that is already available. Libraries like-:

- CV2(OpenCV)
- Matplotlib
- NumPy

These 3 libraries are mainly used in object detection. Cv2 is used to train the model and build the model for object detection, Matplotlib is used to print the output video or image on which object must be detected, NumPy is used to perform the complex mathematical operations on the arrays or other mathematical data.

### 5.2. Importing necessary files

There are various files that are to be imported in the project for effective and accurate prediction of the image. We must import 3 files:

- Config file / mobile net SSD file[6]
- Frozen model[7]
- Coco dataset[8]

We have to import the config file as it contains the mobile net SSD algorithm that will help to predict the objects in the image, frozen model contains the frozen inference graph that used to trained deep learning model that has been optimized and converted into suitable format for inference, and coco dataset is a file that contain the labels of all the different classes that could be predicted.

### 5.3. Pre-Processing

We must apply pre-processing techniques to enhance the quality of the video frames, this includes resizing, normalizing, denoising and colour space conversion. We must first create an array of all the

DOI: 10.5281/zenodo.10935287

different classes of the coco dataset. Furthermore, we do this by applying splitting the input label file and transferring the independent labels like people, car, knife, etc. one by one to the array. In addition, we must alter the input frame size to 320*320 and alter the scale of the frames of the video to 1.0/127.5 we chose 127.5 as the half of 255 is 127.5 and 255 is the total component in RGB component. We must set the input mean the of all the RGB component to 127.5, and then we have to swap the BGR (blue, green, red) to the RGB (red, green, blue) of the image.

### 5.4. Object Detection Model

We must select the pre-trained Object detection model that suits the requirements. We will use the DNN model of the OpenCV, here we must use the dnn_Detectionmodel() and we use frozen model, config_file as the parameters of the function that we are using for creating and training the detection model.

### 5.5. Model Initialization

We must load the pre-trained model using OpenCV's DNN module. We have to set up the network by loading the model's architecture and weight. We use a variable called "image_test". We initialized the model with the help of the "imread" module of the CV2 library, here we must input the test image as the parameter for the dnn model.

### 5.6. Video Frame Processing

Now we have to process the video so that object detection can be done on it, we have to understand that video is just a sequence of frames/images that are being displayed in a continuous and fast rate. Now we have to split up all the frames of the video and then iterate though all the frames and apply the pre-trained object detection model. We have to convert the frame into a blob for input to the model. Forward pass the blob through the network to obtain the detection. This is where we find python helpful as the CV2 library help this task of splitting the frames very easily and with very fewer lines of code. Here we also define the threshold value and confidence value which are helpful in detecting the object.

Confidence value is the mathematical value or score that shows the probability of the image being detected correctly by the algorithm and is given as a percentage score.

Threshold value is the mathematical value that is detected either manually or automatically. If the value of confidence for any object is above the threshold value, then the object class is said to be detected in that particular area.

Also, we have to flatter the intermediate result that we have got after the processing i.e., the result that we have got might be in 2 dimension or higher dimensions, so we have to reduce its dimensionality to a single dimension array of classes of the objects. Finally, we will iterate all the frames of the video until no more frames are left to be processed, i.e., till the end of the video.

### 5.7. Post-Processing

Now, after the processing of the image is done, we have to perform certain operations to analyze the detection result obtained from the model. We filter the detections based on confidence score to remove low confidence detection. We also set up the box size that is to be used to detect all the distinct objects and the colour of the box, border size of the box and at the same time text formatting is done. The size of the text that is to be displayed is selected, and the font size is also selected.

### 5.8. Visualization

Now we use Matplotlib library[9] of python to visualize the image or video on which the object detection has been done. Here the boundary boxes are drawn, and other relevant information are provided with the video frames. In this step we show or save the processed video frames with the detected objects so that further processing can be done on the input video, or the video can be shown as an output. In this step we use the <PLT.IMSHOW> command, the <imshow> is the module of the Matplotlib library and here we pass the input image/video, and the command to convert the BGR scale to RGB scale if it has

DOI: 10.5281/zenodo.10935287

not been done earlier and also, we could give some sample text as the title or heading of the video and format (optional), as parameters.

### 5.9. Output Analysis and Refinement

This is the last step that has to be performed. In this, we have to review the detected objects as shown in fig. 2 and Fig. 3 in the video frames and evaluate the accuracy and consistency of the model that we have built. We also have to perform the identification of any false positive or missed detection. We have to refine the model by adjusting the parameters of the model such as confidence value, threshold value or NMS threshold value. We have to iterate the refinement step to find for what values of the parameters, the accuracy and consistency of the model is coming the best.

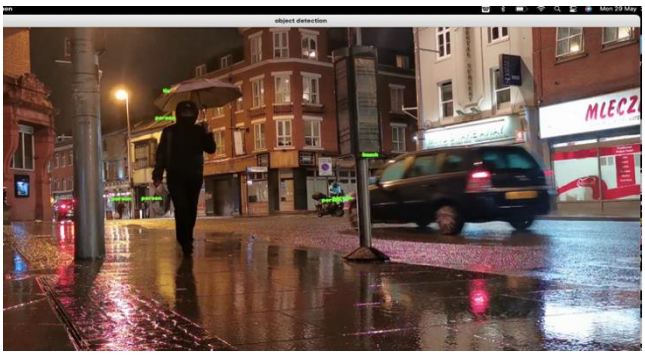


**Figure 2:** Detecting Multiple Persons

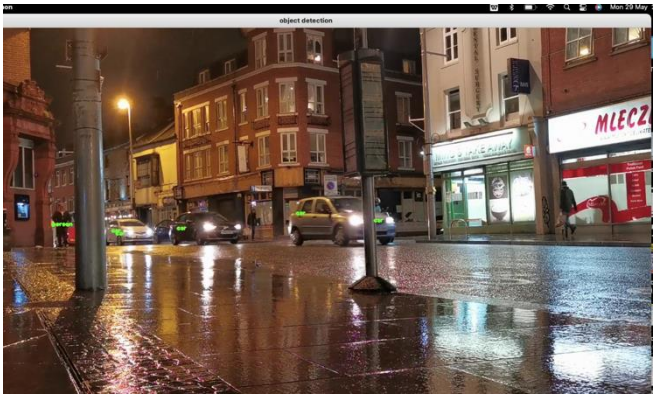


**Figure 3:** Detecting Multiple objects like cars and person.

## 6. EVALUATION

### 6.1. Accuracy

OpenCV can achieve the accuracy level of above 60%. But in this case, we have used mobile net SSD algorithm over the OpenCV. Mobile net SSD is a powerful algorithm for object detection; hence it increases the overall accuracy of the model, and we finally get the average accuracy of the model as 89.2%. It is to be noted that this accuracy has been achieved with the video that has been given as input to the model and not to the real time live video captured by the webcam. The accuracy will further increase if we use this model only for object detection in images. The value achieved is well above the industry standard reference.

### 6.2. Detection Speed

The detection speed achieved by the model was somewhere close to 16.2 Fps for object detection on the dataset, which can identify the object detection quickly and accuracy the value achieved by the model is well above the set industry standard/reference.

DOI: 10.5281/zenodo.10935287

### 6.3. Parameter of the model (Threshold value)

The threshold value of the model is the set value. If the confidence value of any object crosses the threshold value, then that object is said to be detected in the video. For this model we experimented on a wide range of threshold value, and we got the best results and accuracy at 0.6. Hence, the threshold value of the model was fixed at 0.6.

## 7. CONCLUSION

In conclusion, this project successfully demonstrates the application of object detection techniques using the MobileNet SSD algorithm with OpenCV in Python. Object detection is a fundamental computer vision task with numerous practical applications in various fields, including surveillance, autonomous vehicles, robotics, and more. The project showcases the capability of accurately identifying and localizing multiple objects within images and videos, leveraging the power of deep learning and pre-trained models.

The combination of OpenCV and MobileNet SSD proves to be a powerful solution for object detection, achieving a remarkable accuracy level of 89.2%. This high accuracy is a result of the sophisticated architecture of the MobileNet SSD algorithm, which efficiently balances speed and precision. The model's ability to process video frames at a speed of 16.2 FPS ensures real-time object detection, making it applicable to dynamic scenarios.

Moreover, the project highlights the importance of dataset selection, using the widely known COCO dataset, which includes a diverse set of 80 object categories, to train the model effectively. The availability of pre-trained models and the ease of integration with OpenCV enable developers to save time and resources when implementing object detection systems.

The post-processing step ensures proper visualization of detected objects by drawing bounding boxes and labels on the image or video frames. The ability to analyse and refine the model based on accuracy and consistency evaluations demonstrates the importance of fine-tuning the parameters to achieve optimal results.

In terms of hardware requirements, the project efficiently runs on a standard Mac machine, demonstrating the feasibility of deploying object detection applications on relatively modest hardware setups.

Overall, the combination of Python, OpenCV, MobileNet SSD, and the COCO dataset provides a robust and accessible platform for object detection tasks. Developers can further extend this work to real-time video streaming from webcam or integrate it into larger systems for various applications. Object detection continues to be a rapidly evolving field, and with the continuous advancements in deep learning and computer vision, we can expect even more accurate and efficient solutions in the future.

The project on object detection in video using OpenCV has demonstrated the effectiveness and versatility of this approach in various real-world applications. By leveraging the powerful features and algorithms provided by OpenCV, the project successfully implemented an object detection system that can accurately identify and track objects in video streams.

The methodology employed in the project involved several key steps, including video pre-processing, feature extraction, and object detection using pre-trained deep learning models. OpenCV's extensive library of functions and algorithms, combined with deep learning techniques, allowed for efficient and accurate object detection in videos.

Through rigorous testing and evaluation, the project validated the performance of the object detection system. The system exhibited robustness in detecting objects of interest, providing accurate bounding box predictions and classifying various objects effectively. Performance metrics such as Intersection over Union (IOU), Average Precision (AP), and frames per Second (FPS) were used to assess the system's accuracy, precision, and efficiency.

The project demonstrated the wide range of applications for object detection in video using OpenCV. It showcased its potential in fields such as surveillance systems, autonomous vehicles, retail analytics,

DOI: 10.5281/zenodo.10935287

crowd management, and industrial automation. By enabling real-time monitoring and analysis of objects in video streams, the system enhances situational awareness, improves security measures, and supports decision-making processes.

Furthermore, the project highlighted the significance of OpenCV as a versatile and powerful computer vision library. It provided a comprehensive set of tools and functions for video pre-processing, feature extraction, and object detection, simplifying the implementation of the model/process.

To sum-up, the project on object detection in video using OpenCV has showcased the effectiveness, versatility, and potential of this approach in various domains. With further advancements in computer vision and deep learning, object detection in video using OpenCV is expected to continue driving innovations and finding practical applications in numerous fields, contributing to enhanced surveillance, automation, and decision-making

Furthermore, the project highlighted the significance of OpenCV as a versatile and powerful computer vision library. It provided a comprehensive set of tools and functions for video pre-processing, feature extraction, and object detection, simplifying the implementation of the model/process.

To sum-up, the project on object detection in video using OpenCV has showcased the effectiveness, versatility, and potential of this approach in various domains. With further advancements in computer vision and deep learning ,object detection in video using OpenCV is expected to continue driving innovations and finding
practical applications in numerous fields, contributing to enhanced surveillance, automation, and decision-making.

## REFERENCES

[1] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, Bo Ma Reserch on YOLO A Review of Yolo Algorithm Developments volume 199, 2022, 1066-1073

[2] Ryan Nash, Kelron Tello O'Shea Reserch on An Introduction to Convolitional Neural Networks.2015

[3] Souhail Guennouni, Ali Ahaitouf and Anass Mansouri Reserch on Multiple Object Detection using OpenCV on an Embaded Platform. 2015, 374: 8:9

[4] R.B Research on Application of Intelligent Video Surveillance and Face Recognition Technology in Prision Security. China Security Technology and Application. 2019,6: 16-1.

[5] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen Research on Tensorflow: A system for Large-Scale Machine Learning.

[6] Data available at: https://github.com/tensorflow/models/blob/master/research/object_detection/samples/configs/ssd_mobilenet_v1_pets.config

[7] https://www.kaggle.com/code/chienhsianghung/object-detection-using-opencv-inference

[8] https://www.kaggle.com/datasets/awsaf49/coco-2017-dataset

[9] Arnav Oberoi and Rahul Chauhan research on Visualizing data using Matplotlib and Seaborn libraries in Python for data science. March 2019 (IJSRP) 9(3):p8733.

## AUTHORS

**Neha Gupta** is an assistant professor in Computer Science and Engineering Department of Moradabad Institute of Technology affiliated with Dr. A.P.J. Abdul Kalam Technical University. She had done B. Tech, M.Tech and now pursuing Ph. D. Her research area involves Machine learning, Deep Learning, Data Science and Data Security Measures.

**Prakhar Kapoor** is a B.Tech 3rd Year Student in Computer Science and Engineering Department of Moradabad Institute of Technology affiliated with Dr. A.P.J. Abdul Kalam Technical University. His research interests include Machine Learning, Deep learning and Video data processing.

DOI: 10.5281/zenodo.10935287

**Sejal Thakur** is a B.Tech 3rd Year Student in Computer Science and Engineering Department of Moradabad Institute of Technology affiliated with Dr. A.P.J. Abdul Kalam Technical University. Her research interests include Machine Learning, Deep learning and Video data processing.

DOI: 10.5281/zenodo.10935287