IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876

*Research paper*  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

# Object Detection and Screen Presence Time Estimation Using Opencv and Yolo Alogrithm

**DOI:10.48047/IJFANS/V11/I12/190**

**K. Mohan Krishna**[1], Associate Professor, Department of CSE,
Vasireddy Venkatadri Institute of Technology, Nambur, Guntur Dt., Andhra Pradesh.

**K Akhil Chowdari**[2], **J Anusha**[3], **J V N S L Sravani**[4], **K Sreya Chowdary**[5]
[2,3,4,5] UG Students, Department of CSE,
Vasireddy Venkatadri Institute of Technology, Nambur, Guntur Dt., Andhra Pradesh.
**[1] E-Mail ID: mohanakrishnakotha@vvit.net, koppakaakhil2002@gmail.com, anushajutke123@gmail.com, jangalapallisravani@gmail.com, sreyakota1928@gmail.com**

**Abstract**

This paper describes about the object detection and object's screen presence time estimating model which will run on any computer easily. Currently there are only models that will only detect the motion of the object alongside estimating screen time. But here we tried to propose a model that will detect the object that is present in the camera view and also estimate the amount of time the object is present in the camera view. For object detection there are many algorithms such as CNN (Convolutional Neural Network), R-CNN, Fast R-CNN, Faster R-CNN, SSD (Single Shot Detector), YOLO (You Only Look Once) etc. In the current model we gave preference to the speed of detecting the object along with the accuracy. So, we preferred using the Yolo algorithm among all the existing algorithms that can be used for the object detection. But yolo is only designed for using in the GPU based computers. So, in order to implement yolo algorithm in our normal CPU based computers we used OpenCV library such that real time object detection is also possible in Non-GPU computers. Our model estimates the time of screen presence of each object using python libraries such as pandas, time etc.

As we used yolo as our object detection algorithm our model detects objects with 80-99% confidence.

**Keywords:** Object Detection, YOLO, OpenCV, Pandas, Numpy

## Introduction

Object detection is a computer vision technique for locating instances of objects in images or videos. To produce meaningful results, Object detection algorithms typically uses the machine learning or deep learning. When humans look at images or videos, they have the ability to recognize and locate objects of interest easily within the moments of time. The goal of object detection is to provide this intelligence of detecting and locating objects using a computer. This object detection is majorly helping in areas like vehicle detection, CCTV

*Research paper*  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

monitoring and also object detection plays a key role in the self-driving cars etc. As there are many object detecting algorithms which are based on deep learning such as CNN (Convolutional Neural Networks), R-CNN, SSD, YOLO and many others. we can use any one of these algorithms to perform the object detection task.

There are two main strategies to perform object detection using deep learning. They are:

The first approach is that we can create a custom object detector and train it with our own data. In order to train our object detector a network architecture is needed to be designed to learn the features for the required objects to be detected. We also need to define a very large set of labelled data to train our model. The custom object detectors can produce amazing results.

The second approach is that we can also use the pretrained object detecting models. Since these object detectors have already been trained on thousands or millions of images, these models produce the result in the least possible time.

Here in the paper we are proposing a model that will perform the object detection along with the estimation of amount of time the object is being visible in front of the camera. As there are many methods and algorithms that are available for the object detection which we have discussed above, here we wanted to use the YOLO (You Only Look Once) object detection algorithm which is the fastest, efficient and accurate algorithm with good mAP (mean average precision). But as yolo is a deep learning approach of detecting the objects it can be only implemented in GPU based computers. So, in order to implement the yolo algorithm in the Non-GPU based computers, we used OpenCV, a popular python library to implement the yolo algorithm in our model. Among the two available strategies of object detection using deep learning, we used a pretrained yolo model which uses a weights file that contains the data of the objects such as bicycle, person, train, car, bottle and many other.

Here the input for our model is taken by using a static web camera. Our model can be used as a proctoring tool that detects the person or objects that appears during the online exam and estimates the amount of time each object is present in front of the camera. The output details of our model will be recorded and stored in a csv file using python libraries such as time, pandas etc.

In this paper we tried to propose a model that can be used by anyone easily and can be easily integrated with any examination conducting software to increase the proctoring features of the portal.

*Research paper* © 2012 IJFANS. All Rights Reserved, UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

**Literature Survey**

Authors Renuka raut, Hirendra hazare proposed a paper in which they describe about performing object detection using Haar feature based cascade classifiers using OpenCV [1]. Every haar classifier needs to be trained with lot of positive and negative images. There are many pre trained classifiers for face, eyes, smile etc. In this paper authors used these pre trained classifiers for the detection of eyes and faces of persons in the given images or videos. The accuracy of a haar classifier depends on the dataset that is trained. The usage of haar classifier takes lot of time for the detection of objects.

Ankita Rameshwar Mahajan, Vinod Agarwal proposed a paper in which they monitor a person's movements and fires an alarm if the person makes any movement and the model also counts the person's movements [2]. In this paper, authors used OpenCV library for the detection of object's movement and the movements of the object is counted by the tracker and finally win sound library is used for making the sound if any movement is detected. The authors suggest that this model can be implemented easily with low cost and with low storage and the results are accurate up to 95%.

Kushal M, Kushal Kumar B V, Charan Kumar M J proposed their research work in which they performed Object detection using the tensor flow object detection API that detects and recognizes the face using haar cascade method of OpenCV [3]. The usage of tensor flow for the object detection requires collection of various images as dataset and need to be labelled and trained and finally detection of objects. In this paper the authors used this object detection technique to detect whether a person wears an ID card or not and detects the face of the person if the person doesn't wear any ID card. The model proposed in this paper is not able to detect the moving objects accurately.

Authors Chandan G, Ayush Jain, Harsh Jain proposed a research paper in which they performed object detection using one of the popular object detection algorithms named SSD which is very accurate and efficient [4]. Besides the presence of many object detection algorithms, authors preferred to implement SSD in OpenCV to detect and track the objects. The objects that are needed to be detected by the model are needed to be trained in prior to the detection.

Author Md. Bahar Ullah proposed a paper that describes about the implementation of famous object detection algorithm named YOLO using OpenCV. As yolo was only designed to implement only on GPU based computers this research paper proposed a model that optimizes YOLO with OpenCV in a way that the detection of objects can be possible on CPU based computers also. As YOLO is one of the fastest algorithms the proposed model will give the results in the least possible time when compared to the usage of other algorithms. The proposed model in this paper has a confidence value of 80-99%.

*Research paper*  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

**Problem Identification**

The objective of this project is to develop a system that can detect the presence of specific objects in a video feed and estimate the amount of time that each object is present on the screen. The system should use OpenCV to capture video from a camera or a pre-recorded video file, and it should use YOLO to detect the presence of specific objects in each frame of the video. Once an object is detected, the system should start a timer to track the object's presence time. When the object is no longer present, the system should stop the timer and calculate the total time that the object was on the screen. The results of the screen presence time estimator should be displayed in a user-friendly format, either overlaid on the video feed or output to a text file or a database. The system should be able to detect multiple objects simultaneously and provide accurate presence time for each object. The system should be designed to be scalable and able to handle real-time video feeds with a high degree of accuracy.

**Methodology**

Here is a high-level methodology for building an object detection and screen presence time estimator using OpenCV and YOLO:

1. Install OpenCV and YOLO: You'll need to install OpenCV and YOLO on your computer to begin the process. OpenCV is an open-source computer vision library, and YOLO (You Only Look Once) is a popular object detection algorithm.
2. Collect Data: Collect a set of images or videos that will be used to train the object detection model. These images should include the objects that you want to detect, and they should be labeled with bounding boxes around the objects.
3. Train the Object Detection Model: Use YOLO to train a custom object detection model using the labeled data. This model should be trained to recognize the specific objects you want to detect.
4. Build a Screen Presence Estimator: Once you have the object detection model, you can use it to track the presence of specific objects in a video feed. You can use OpenCV to capture video from a camera or from a pre-recorded video file. Then, you can use the object detection model to detect the presence of specific objects in each frame of the video.
5. Estimate Screen Presence Time: After you've detected the presence of specific objects in the video, you can calculate the total time that each object is present on the screen. You can do this by keeping track of the time when the object first appears in the video and the time when it disappears. Then, you can calculate the difference between these two times to get the object's total presence time.

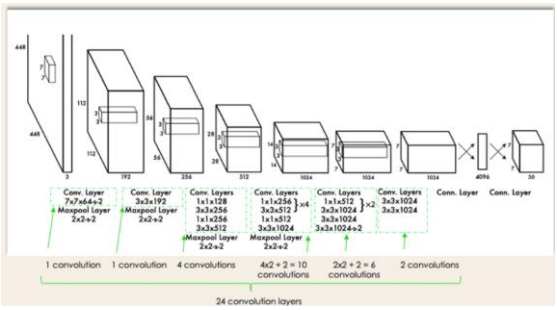
*Research paper*  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

6. Display Results: Finally, you can display the results of the screen presence time estimator in a user-friendly format. You can use OpenCV to overlay the results on top of the video feed, or you can output the results to a text file or a database.

**System Implementation**

You Only Look Once (YOLO) is a popular object detection algorithm that proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. YOLO conducts all of its predictions using a single fully connected layer. While YOLO only requires one iteration, methods that use region proposal networks require numerous iterations for the same image. The YOLO algorithm uses a standard deep convolutional neural network to identify objects in an input image.

The architecture of YOLO algorithm is as follows:



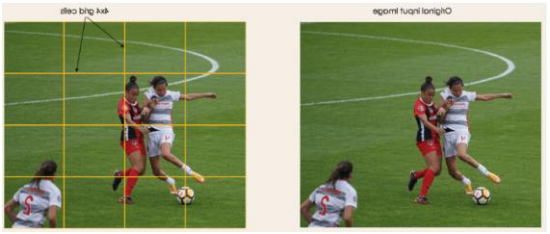

The detection network has 24 convolution layers followed by 2 fully connected layers.

**Working of YOLO Object detection: -**

The yolo algorithm works in the following steps: -

Step 1: Residual Blocks

In this initial step, the algorithm takes an input image and this image is divided into NxN grid cells with equal shapes, where N is 4 in this example as indicated on the right image. The class of the object that each grid cell covers must be predicted locally, together with the probability/confidence value.



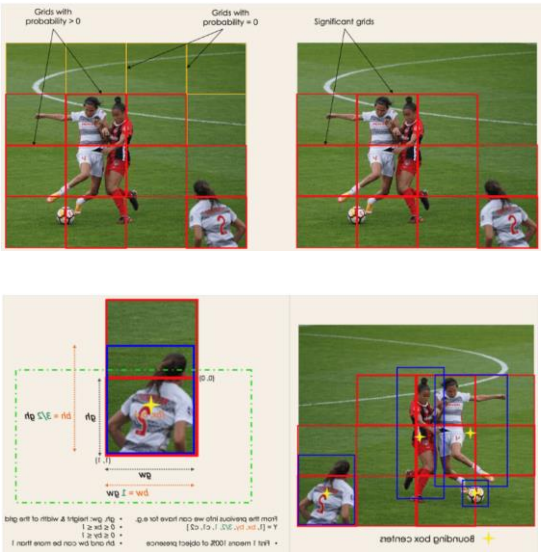
Step 2: Bounding Box Regression

The next step is to identify the bounding boxes, which are rectangles that highlight all of the image's objects. As many bounding boxes as there are objects in a given image is

Research paper  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

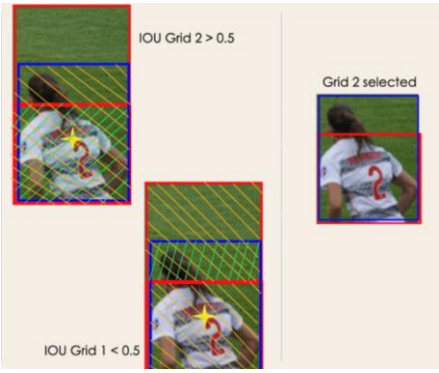
possible. The properties of these bounding boxes are determined by YOLO using a single regression module, where Y is the final vector representation of each bounding box.

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




Step 3: Intersection over Unions (IOU)

A single object in an image might usually have many grid box possibilities for prediction. The IOU's purpose is to eliminate such grid boxes and retain only the necessary ones. The IOU selection threshold is set by the user and can be, for example, 0.5. The IOU, which is the Intersection Area divided by the Union Area, is then calculated for each grid cell by YOLO. Finally, it considers grid cells with an IOU > threshold rather than those predicted to have an IOU >= threshold.




Step 4: Non-Maximum Suppression

After the predictions are made for each cell, the algorithm applies non-maximum suppression to remove overlapping bounding boxes. This step ensures that each object is detected only once, even if it appears in multiple cells or multiple boxes within a cell.

*Research paper*  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

Step 5: Final Output

The final output of the algorithm is a set of bounding boxes, each associated with a class label and a confidence score. These boxes represent the detected objects in the input image.

**Implementation of YOLO algorithm in our project**

The python implementation of yolo involves the following steps:

1. Initially we need to import the libraries.
2. Then, we need to load the yolo model using the readNet() function of OpenCV's deep neural network (dnn) module. This function reads a deep learning network model file and its corresponding configuration file. The function returns a network object that can be used for further processing such as inference and fine-tuning.
   The syntax of the function is
   cv2.dnn.readNet(model, config=None, framework=None)

   model: The path to the file that contains the trained weights of the network. This file should have a .pb, or .t7 extension. Here, we used yolov3.weights file.

   config: The path to the file that contains the network configuration or architecture. Here, we used yolov3.cfg file.

   framework: The name of the deep learning framework used to train the model.
3. The next step is to load the coco.names file and store it as a list.
4. Then the user needs to load the input image and get its dimensions.
5. We then create an input blob from the input image that we have loaded using blobFromImage function of OpenCV's dnn module.
6. We then set blob as the input for the network using the setInput() method of the network object.
7. Then we need to perform a forward pass through the network with the input set using the setInput() method which involves propagating the input through the network layers and computing the output. The output from the forward pass is a 4D numpy array that contains the detection results.
8. Then we need to draw the bounding boxes and display the class names and confidence values using the above results which will be the final result. This can be done using the NMSBoxes, rectangle, putText functions of OpenCV.

Here, the input for our yolo model will be received from the interface file we have designed to make the model to be useful for everyone easily. Using this interface file we will get the input from the web camera which will be divided into frames and will be sent to our yolo

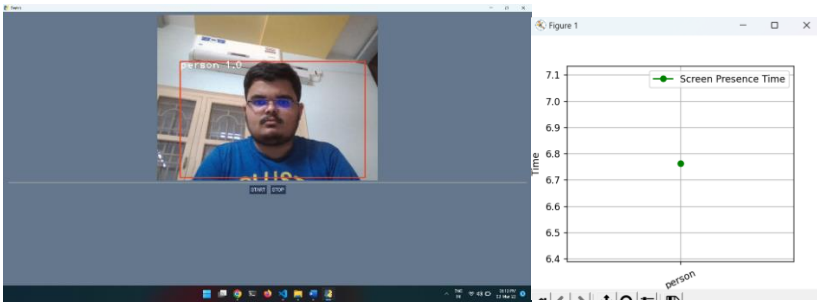
ISSN PRINT 2319 1775 Online 2320 7876

*Research paper*  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022

model. Then our model performs the object detection and returns the output to the interface file where we will display it on the screen.

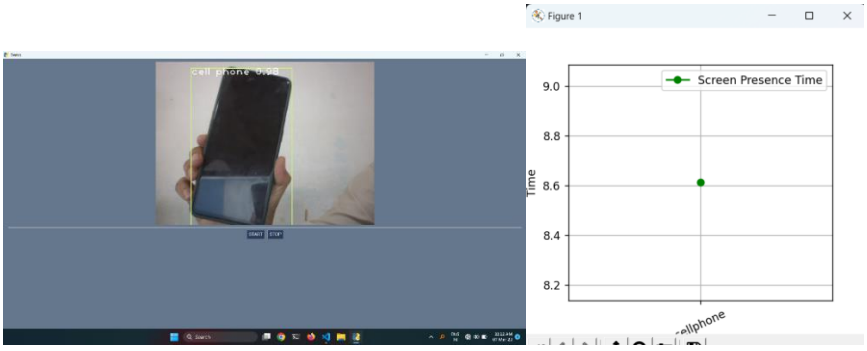
**Results**

Our Model gives us the following results for each of the objects.



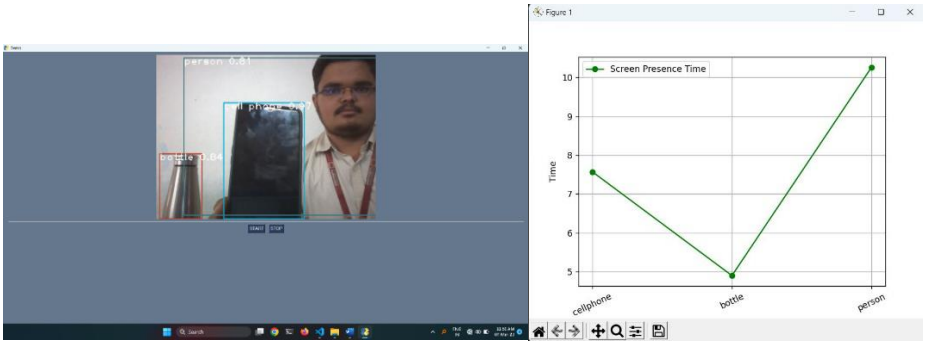

The person is detected with 80 to 100 percent accuracy.


In case of the bottle the accuracy is about 70-95 percent.




In case of a cell phone the model detects the cell phone with 95-100 percent accuracy.

*Research paper*  UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022




If several objects of single instances are captured at a time, then our model will detect these objects and record the screen presence time of each object.

If any object is appeared as multiple instances then our model will detect the objects but the estimation of screen presence time is not done correctly.

**Conclusion**

In conclusion, our research paper demonstrates the development of an Object Detection and Screen Presence Time Estimator system using OpenCV and YOLO. The proposed system is capable of detecting and tracking multiple objects in real-time and estimating the screen presence time of each object in a video stream.

The use of YOLO algorithm provides accurate and efficient object detection, while the OpenCV library offers a powerful set of computer vision tools for image and video processing. By combining these technologies, we have developed a system that can provide valuable insights into the behavior of objects in a video stream.

The experiments conducted in this research paper show that our proposed system is capable of achieving high accuracy in object detection and screen presence time estimation. We have also presented a user-friendly interface that allows users to visualize the results of the detection and tracking process. Our model is able to process the videos with minimum of 10 FPS, and 83-98% confidence within low cost and less effort.

With further development and refinement, this system can be extended to various industries and domains, providing valuable insights and improving user experiences.

**Limitations and Future Scope:**

Limitations:

Single Object's screen presence time estimation: The system is able to estimate the screen presence time of singular objects. That means there should be only one object of each type in the camera frame at a time.

*Research paper* **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed ( Group -I) Journal Volume 11, Iss 12, Dec 2022**

Limited scope: The system is designed to detect specific objects and estimate their presence time. It may not be able to detect other objects or provide information about their presence time.

Future Scope:

Improved object detection accuracy: Future research can focus on improving the accuracy of object detection algorithms, especially in challenging scenarios such as crowded scenes or low-contrast objects.

Extension to other applications: The technology used in this project can be extended to other applications such as traffic monitoring, surveillance, and crowd analysis. The system can also be enhanced to detect and track multiple objects simultaneously and to provide more detailed analysis of object behavior.

**References**

[1] Renuka Raut, Hirendra hazare, "Object Motion Detection and Tracking for Image Surveillance", IJARCCE, Vol. 9, Issue 5, May 2020

[2] Ankita Rameshwar Mahajan, Vinod Agarwal, "Motion Detection using OpenCV", IRJMETS, Vol. 4, Issue 5, May 2022

[3] Kushal M, Kushal Kumar B V, "ID Card Detection with facial Recognition using Tensorflow and OpenCV", IEEE International Conference on Inventive Research in Computing Applications (ICIRCA) , 2020, India.

[4] G Chandan; Ayush Jain; Harsh Jain; Mohana, "Real Time Object Detection and Tracking Using Deep Learning and OpenCV", IEEE International Conference on Inventive Research in Computing Applications (ICIRCA), 2018, India.

[5] Md. Bahar Ullah, "CPU Based YOLO: A Real Time Object Detection Algorithm", IEEE (TENSYMP), 2020, Bangladesh

[6] Maiya Harib Said Al-Hajri, Bushra Al Balushi, Sruthi Mohanan Nambiar, Swati Kishore Pai, Sanjay Kumar, "MOTION DETECTION USING WEBCAM", Issue 5 May, 2014

[7] Irfan Kilic; Galip Aydin, "Traffic Sign Detection And Recognition Using TensorFlow' s Object Detection API With A New Benchmark Dataset", IEEE International Conference on Electrical Engineering (ICEE), 2020, Turkey

[8] Suraiya Parveen; Javeria Shah, "A Motion Detection System in Python and Opencv", IEEE, International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021, India