

## **REAL-TIME MULTIPLE OBJECT DETECTION AND TRACKING**

Nancy, MD Kaif, Ishika Banga , Prince Kumar , Prerna Sikarwar

**Abstract** - In many computer vision applications, such as surveillance, vehicle navigation, and autonomous robot navigation, object recognition and tracking are critical and difficult problems. One of the flow testing research subjects in PC vision is video observation in a powerful climate, particularly for people and cars. It is a critical innovation in the fight against illegal intimidation, crime, public safety, and effective traffic control. The endeavour entails developing an effective video surveillance system for use in complex contexts. Detecting moving things from a video is critical for object identification and target tracking in video surveillance. The detection of moving objects in video streams is the first significant phase of information, and background subtraction is a common method for foreground segmentation. Due to an increased demand for such systems in public spaces such as airports, subway stations, and mass events, intelligent and automated security surveillance systems have been an important study topic in recent years. In this context, one of the most significant needs for surveillance systems based on the tracking of abandoned, stolen, or parked vehicles is the tracking of fixed foreground regions. Because they work reasonably well when the camera is stationary and the change in ambient lighting is gradual, object tracking-based techniques are the most popular choice for detecting stationary foreground objects. They are also the most popular choice for separating foreground objects from the current frame.

## INTRODUCTION

Object detection is the process of locating things in video frames. Tracking is the practise of utilising a camera to locate moving items or several things over time. The difficulty of determining the trajectory or route of an item in the image plane as it moves across a scene is known as tracking. Object tracking algorithms have sparked a lot of interest due to high-powered computers, the availability of high-quality and inexpensive video cameras, and the growing need for automated video analysis. Image classification where we estimate the class or kind of the object seen in the image - is one of the three main tasks we'll use here. Object localization identifies the exact location of an object in a picture. Object detection entails enclosing the object in the image with bounding boxes and determining the sort of object found in the picture. However, because to the wide range of views, positions, occlusions, and lighting conditions, accurate object detection with an extra object localization duty is tough. Object detection is defined as determining where objects are situated in a given image (object localization) and which category each object belongs to, which has gained a lot of interest in recent years (object classification).

## LITERATURE SURVEY

**Paper 1: A real-time object detection algorithm for video:**

**Published year: 23 May 2019**

Unlike the rapid RCNN, YOLO does not divide the function of perceiving an item into different processes, such as object prediction and phase prediction for the region. To accomplish speedy detection with high accuracy, the YOLO method combines both of these functions into a single

neural network model. YOLO stands for "You Only Live Once "sophisticated CNN that claims to employ a fully integrated convolutional neural network to extract features. To predict object details, add a layer. The model has 24 layers of specification and two fully connected layers. Car tracking films from smart cities from

the Xiamen municipal transport office are included in the data sets. That I can only tell the difference between five items. TensorFlow is used to implement the algorithm in Python framework. A Nvidia GPU is required.

**Paper 2: You Only Look Once: Unified, Real-Time Object Detection Author:** Joseph Redmon, Santosh Divvala , Ross Girshick, Ali Farhadi **Published year:** 2018

To locate it, the model is employed as a CNN and evaluated on the PASCAL VOC database. The first few layers of network connectivity are used to locate fully connected features in pictures and layers. Two fully connected layers follow the model layers. It's difficult to recognise a little thing, such as a flock of birds because the binding boxes have a limited amount of space perhaps hundreds of similar binds.

**Paper 3: YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers**

**Author:** Rachel Huang, Jonathan Pedoeem, Cuixian Chen

**Published year:** 2018.

YOLO (You Only Live Once) was created to create a one-step discovery and division procedure. The YOLO's rapid architecture can achieve up to 45 frames per second, and the Tiny-YOLO can achieve 244 frames per second. YOLO-LITE is working on a performance that will be able to handle at least ten frames. with a non-GPU PASCAL VOC computer at 30 frames per second (FPS). Shows how deep shallow networks are. having a rapid recovery option that isn't GPU-based To train and test YOLO, a framework built to construct it was employed. models. The training takes place on an alienware aura with an i7 CPU and a Nvidia 1070 GPU. The Dell XPS 13 is used for testing laptop. Because, each grid can only hold two boxes, it's difficult to get anything closer. It's difficult to find little items.

## Methodology

Real-Time Multiple Object Detection and Tracking may seem like a mouthful to the uninformed. However, with a few fantastic libraries at your disposal, the task becomes far easier than it appears. In this post, we'll use one of these Python libraries, OpenCV, to build a generic software that can detect any item in a video feed.

We'll use openCV for Real-Time Multiple Object Detection and Tracking.

### openCV

OpenCV is a free and open-source library for computer vision problems. The quickest approach to install OpenCV to python is using pip, assuming you have python 3 pre-installed on your PCs.

You can do so by entering the command line below into your command prompt.

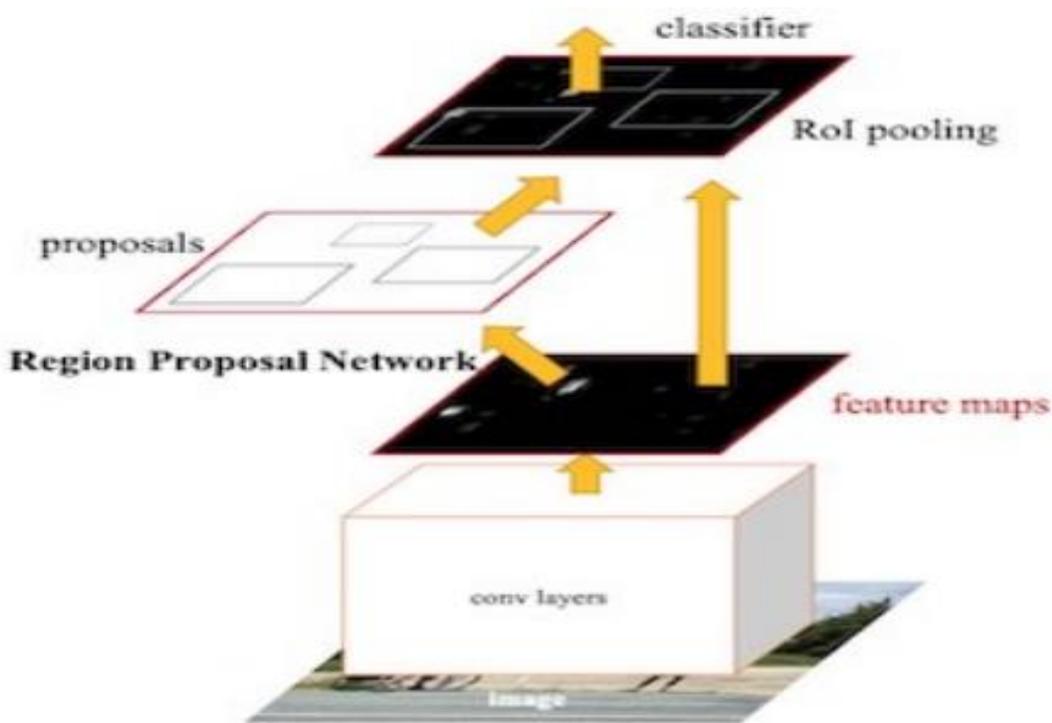
```
pip install opencv
```

### Basic Structure

Object detection models based on deep learning usually contain two elements. An encoder takes an image and passes it through a series of blocks and layers that learn to extract statistical information that can be used to find and classify items. The encoder's outputs are then given to a decoder, which anticipates each object's bounding boxes and labels. A pure regressor is the most basic decoder. The regressor is linked to the encoder's output and directly predicts the location and size of each bounding box. The model's output is the object's X, Y coordinate pair as well as its extent in the image. This type of model is limited, despite its simplicity. The quantity of boxes must be specified ahead of time. If your image has two dogs but your model was only built to detect

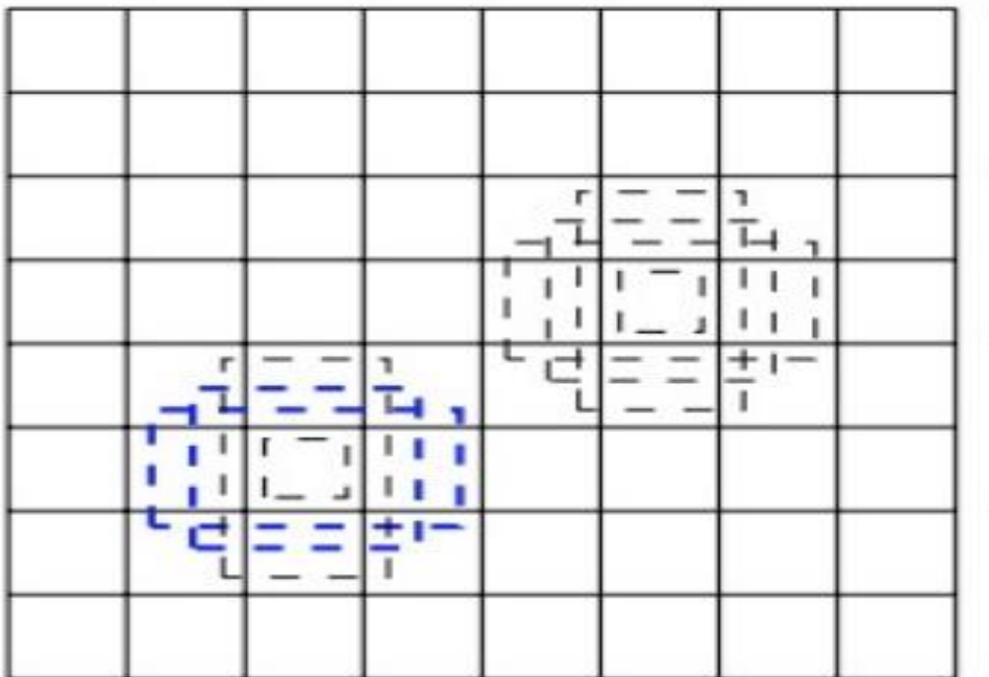
one, one of them will be detected. Pure regressor-based models, on the other hand, may be a viable alternative if you know how many items you need to forecast in each image ahead of time.

A region proposal network is an extension of the regressor technique. The model in this decoder suggests portions of an image where it thinks an object might be. These pixels are subsequently input into a classification subnetwork, which assigns a label to them (or reject the proposal). The pixels containing those regions are subsequently passed through a classification network. The advantage of this strategy is that it produces a more accurate and flexible model that can suggest an unlimited number of regions that could contain a bounding box. However, the increased precision comes at the expense of computing efficiency.

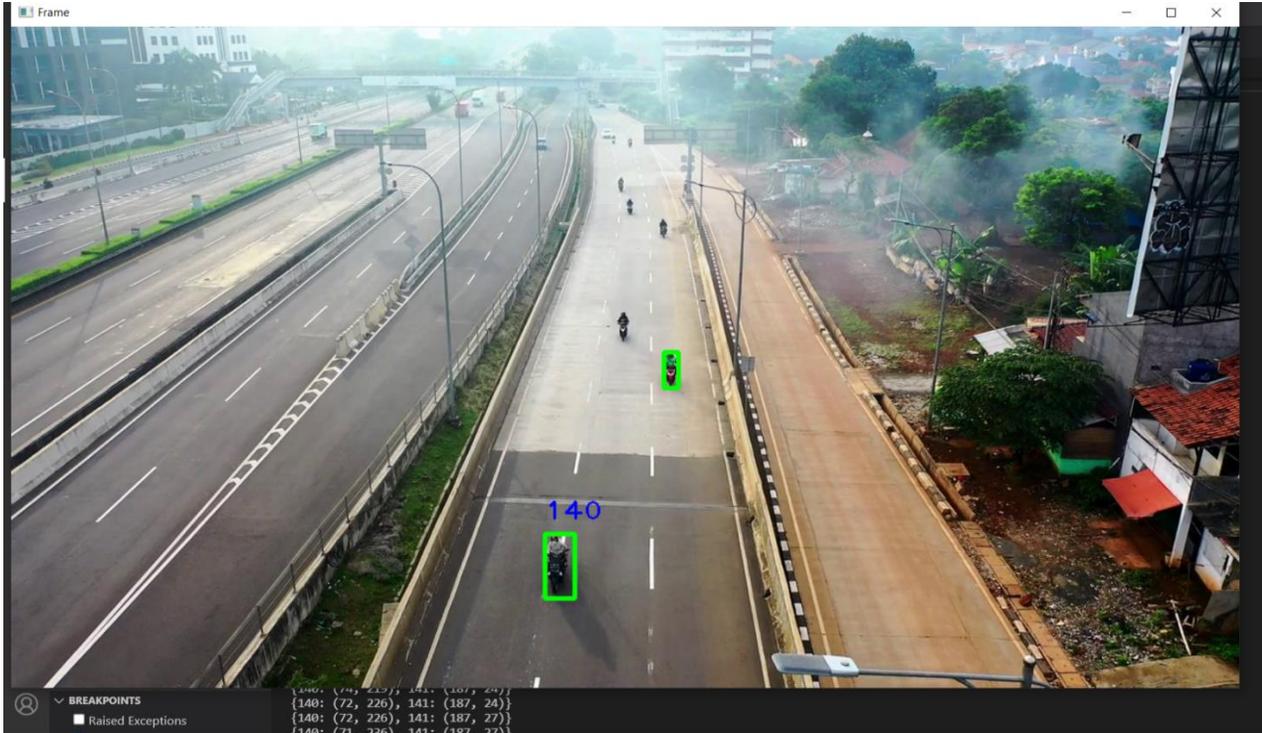


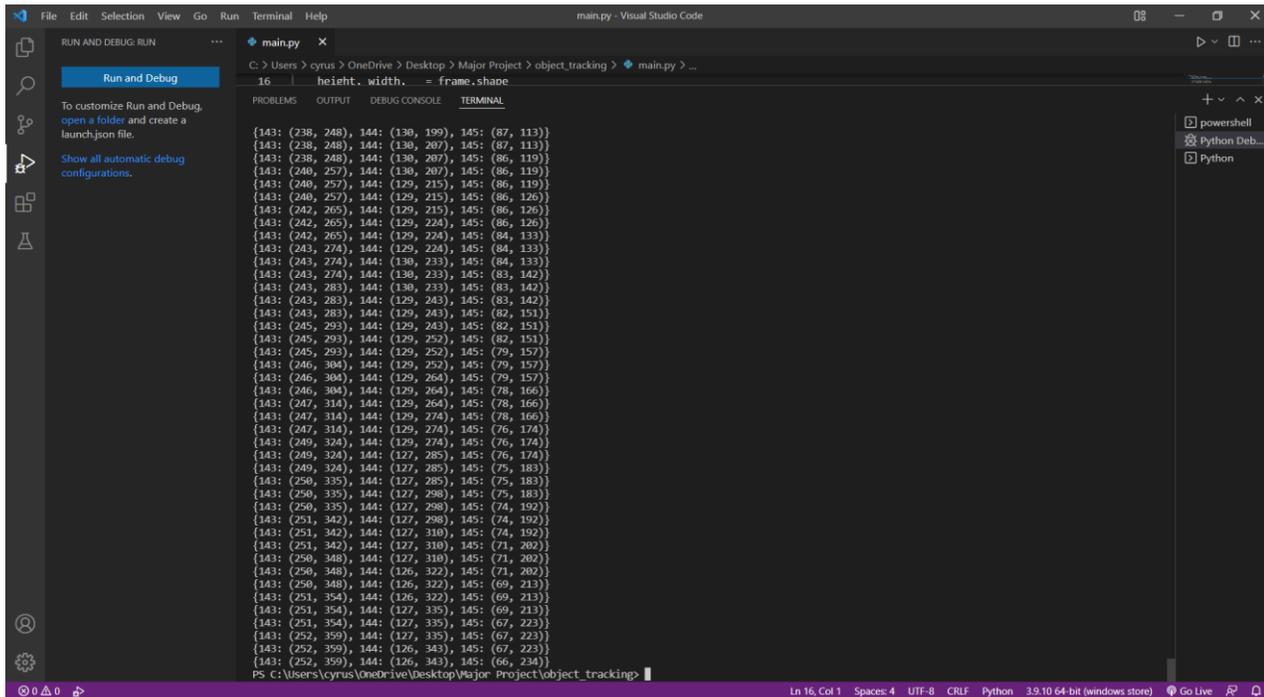
Single shot detectors (SSD) try to find a happy medium. SSD uses a collection of preset areas rather than a sub-network to suggest regions. A grid of anchor points is laid over the input image, with boxes of various shapes and sizes serving as regions at each anchor point. The model provides a prediction of whether or not an object exists within the region for each box at each anchor point, as well as changes to the box's location and size to

better match the object. SSDs produce several potential detections that overlap because each anchor point has multiple boxes and anchor points may be near together. To exclude the majority of these predictions and choose the best one, post-processing must be applied to SSD outputs. Non-maximum suppression is the most widely used post-processing approach.



## RESULTS





```
main.py - Visual Studio Code
C:\Users\cyrus> OneDrive\ Desktop\ Major Project\ object_tracking > main.py > ...
16 | heieht, width, = frame.shape
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
{143: (238, 248), 144: (130, 199), 145: (87, 113)}
{143: (238, 248), 144: (130, 207), 145: (87, 113)}
{143: (238, 248), 144: (130, 207), 145: (86, 119)}
{143: (240, 257), 144: (130, 207), 145: (86, 119)}
{143: (240, 257), 144: (129, 215), 145: (86, 119)}
{143: (240, 257), 144: (129, 215), 145: (86, 126)}
{143: (242, 265), 144: (129, 215), 145: (86, 126)}
{143: (242, 265), 144: (129, 224), 145: (86, 126)}
{143: (242, 265), 144: (129, 224), 145: (84, 133)}
{143: (243, 274), 144: (129, 224), 145: (84, 133)}
{143: (243, 274), 144: (130, 233), 145: (84, 133)}
{143: (243, 274), 144: (130, 233), 145: (83, 142)}
{143: (243, 283), 144: (130, 233), 145: (83, 142)}
{143: (243, 283), 144: (129, 243), 145: (83, 142)}
{143: (243, 283), 144: (129, 243), 145: (82, 151)}
{143: (245, 293), 144: (129, 243), 145: (82, 151)}
{143: (245, 293), 144: (129, 252), 145: (82, 151)}
{143: (245, 293), 144: (129, 252), 145: (79, 157)}
{143: (246, 304), 144: (129, 252), 145: (79, 157)}
{143: (246, 304), 144: (129, 264), 145: (79, 157)}
{143: (246, 304), 144: (129, 264), 145: (78, 166)}
{143: (247, 314), 144: (129, 264), 145: (78, 166)}
{143: (247, 314), 144: (129, 274), 145: (78, 166)}
{143: (247, 314), 144: (129, 274), 145: (76, 174)}
{143: (249, 324), 144: (129, 274), 145: (76, 174)}
{143: (249, 324), 144: (127, 285), 145: (76, 174)}
{143: (249, 324), 144: (127, 285), 145: (75, 183)}
{143: (250, 335), 144: (127, 285), 145: (75, 183)}
{143: (250, 335), 144: (127, 298), 145: (75, 183)}
{143: (250, 335), 144: (127, 298), 145: (74, 192)}
{143: (251, 342), 144: (127, 298), 145: (74, 192)}
{143: (251, 342), 144: (127, 310), 145: (74, 192)}
{143: (251, 342), 144: (127, 310), 145: (71, 202)}
{143: (250, 348), 144: (127, 310), 145: (71, 202)}
{143: (250, 348), 144: (126, 322), 145: (71, 202)}
{143: (251, 354), 144: (126, 322), 145: (69, 213)}
{143: (251, 354), 144: (126, 322), 145: (69, 213)}
{143: (251, 354), 144: (127, 335), 145: (69, 213)}
{143: (251, 354), 144: (127, 335), 145: (67, 223)}
{143: (252, 359), 144: (127, 335), 145: (67, 223)}
{143: (252, 359), 144: (126, 343), 145: (67, 223)}
{143: (252, 359), 144: (126, 343), 145: (66, 234)}
PS C:\Users\cyrus\OneDrive\Desktop\Major Project\object_tracking
```

## Tracking And Counting

## Conclusion

We have presented a comparative overview of the existing work in the field of object detection for visually impaired people in this paper. This investigation reveals that current systems are ineffective and inconvenient for visually impaired people. Existing systems are either ineffective in real-world scenarios or necessitate some level of dependency. The majority of solutions require additional hardware, which adds to the load for visually impaired persons. There is a need for a system that will not feel like a burden to them and will also become a part of their daily lives. This requirement of visually impaired people can be met by our proposed system. Our suggested technology will assist visually impaired people in detecting and identifying objects, as well as informing them about their surroundings. It will assist them by detecting and identifying objects both indoors and outside. Following the object's detection, the system will generate a speech to alert visually impaired persons. We should now be familiar with some of the most common—as well as a number of newer—methods for detecting objects in a range of situations.

### References

- [1]. Mohammad Javad Shafiee, Brendan Chywl, Francis Li, Alexander Wong, Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video
- [2]. Y. Meng, “Agent-based reconfigurable architecture for real-time object tracking,” *Journal of Real-Time Image Processing*, vol. 4, no. 4, pp. 339–351, 2009.
- [3]. C. Tomasi and T. Kanade, “Shape and motion from image streams under orthography: a factorization method,” *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.
- [4]. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi You Only Look Once: Unified, Real-Time Object Detection. [5]. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi You Only Look Once: Unified, Real-Time Object Detection.
- [5]. R. Zabih and J. Woodfill, “Non-parametric local transforms for computing visual correspondence,” in *Computer Vision ECCV’94*. Springer, 1994, pp. 151–158.