TRAFFIC MANAGEMENT

VEHICLE DETECTION

Overview

- Excited by the idea of smart cities? You'll love this tutorial on building your own vehicle detection system
  We'll first understand how to detect moving objects in a video before diving into the implementation part
- We'll be using OpenCV and Python to build the automatic vehicle detector

Introduction

I love the idea of Traffic Management System. The thought of automated smart energy systems, electrical grids, one-touch access ports – it's an enthralling concept! Honestly, it's a dream for a data scientist and I'm delighted that a lot of cities around the world are moving towards becoming smarter.

One of the core components of a smart city is automated traffic management. And that got me thinking – could I use my data science chops to build a vehicle detection model that could play a part in smart traffic management?

Think about it – if you could integrate a vehicle detection system in a traffic light camera, you could easily track a number of useful things simultaneously:

- How many vehicles are present at the traffic junction during the day? What time does the traffic build up?
- What kind of vehicles are traversing the junction (heavy vehicles, cars, etc.)?
- Is there a way to optimize the traffic and distribute it through a different street?
-

And so on. The applications are endless!

Us humans can easily detect and recognize objects from complex scenes in a flash. Translating that thought process to a machine, however, requires us to learn the art of object detection using computer vision algorithms.

So in this article, we will be building an automatic vehicle detector and counter model. Here's a taste of what you can expect:

Excited? Let's turn on the ignition and take this for a spin!

Note: New to deep learning and computer vision? Here are two popular courses to kick start your deep learning journey:

Fundamental of deep learning

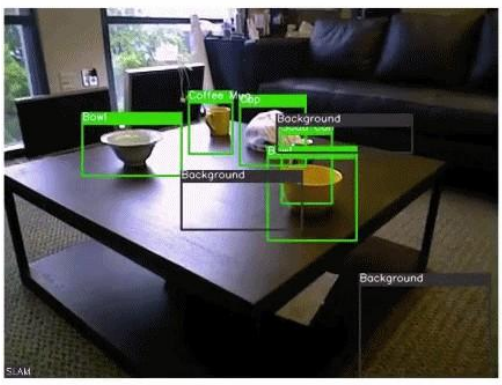
Computer vision using Deep Learning

Table of Contents

The Idea Behind Detecting Moving Objects in Videos

Object detection is a fascinating field in computer vision. It goes to a whole new level when we're dealing with video data. The complexity rises up a notch, but so do the rewards!

We can perform super useful high-value tasks such as surveillance, traffic management, fighting crime, etc. using object detection algorithms. Here's a GIF demonstrating the idea:



There are a number of sub-tasks we can perform in object detection, such as counting the number of objects, finding the relative size of the objects, or finding the relative distance between the objects. All these sub-tasks are important as they contribute to solving some of the toughest real-world problems.

If you're looking to learn about object detection from scratch, I recommend these tutorials:

- Step by step introduction to the Basics object Detection .
- Real time object Detection using slimYOLOv3
- Other objects Detection Articles and resources

Let's look at some of the exciting real-world use cases of object detection.

Real-World Use Cases of Object Detection

Nowadays, video object detection is being deployed across a wide range of industries. The use cases range from video surveillance to sports broadcasting to robot navigation.

Here's the good news – the possibilities are endless when it comes to future use cases for video object detection and tracking. Here I've listed down some of the interesting applications:

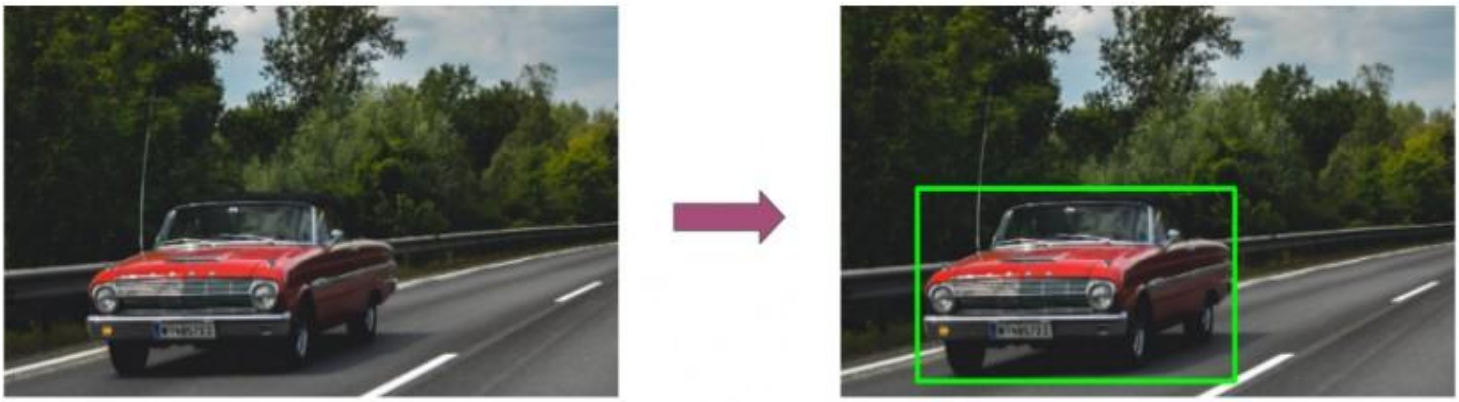
1. Crowd Counting
2. Vehicle number plate detection and recognition
3. Ball Tracking
4. Robotics
5. Traffic management (an idea we'll see in this article)

Essential Concepts you should know about Video Object Detection

There are certain key concepts you should know before getting started with building a video detection system. Once you are familiar with these basic concepts, you would be able to build your own detection system for any use case of your choice.

So, how would you like to detect a moving object in a video?

Our objective is to capture the coordinates of the moving object and highlight that object in the video. Consider this frame from a video below:



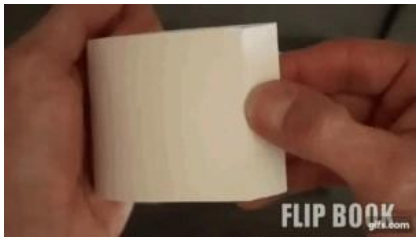
We would want our model to detect the moving object in a video as illustrated in the image above. The moving car is detected and a bounding box is created surrounding the car.

There are multiple techniques to solve this problem. You can train a deep learning model for object detection or you can pick a pre-trained model and fine-tune it on your data. However, these are supervised learning approaches and they require labeled data to train the object detection model.

In this article, **we will focus on the unsupervised way of object detection in videos, i.e., object detection without using any labeled data**. We will use the technique of **frame differencing.** Let's understand how it works!

Frame Differencing

A video is a set of frames stacked together in the right sequence. So, when we see an object moving in a video, it means that the object is at a different location at every consecutive frame.



If we assume that apart from that object nothing else moved in a pair of consecutive frames, then the pixel difference of the first frame from the second frame will highlight the pixels of the moving object. Now, we would have the pixels and the coordinates of the moving object. This is broadly how the frame differencing method works.

Let's take an example. Consider the following two frames from a video:

Frame 1

Frame 2

Can you spot the difference between the two frames?

Yes – it is the position of the hand holding the pen that has changed from frame 1 to frame 2. The rest of the objects have not moved at all. So, as I mentioned earlier, to locate the moving object, we will perform frame differencing. The result will look like this:

You can see the highlighted or the white region where the hand was present initially. Apart from that, the notepad is also highlighted a bit along its edges. This could be due to the change in the illumination by the movement of the hand. It is advisable to get rid of unwanted detection of stationary objects. Therefore, we would need to perform certain image pre-processing steps on the frames.

Image Thresholding

In this method, the pixel values of a grayscale image are assigned one of the two values representing black and white colors based on a threshold. So, if the value of a pixel is greater than a threshold value, it is assigned one value, else it is assigned the other value.

In our case, we will apply image thresholding on the output image of the frame differencing in the previous step:

You can see that a major part of the unwanted highlighted area has gone. The highlighted edges of the notepad are not visible anymore. The resultant image can also be called as a binary image as there are only two colors in it. In the next step, we will see how to capture these highlighted regions.

Finding Contours

The contours are used to identify the shape of an area in the image having the same color or intensity. Contours are like boundaries around regions of interest. So, if we apply contours on the image after the thresholding step, we would get the following result:

The white regions have been surrounded by grayish boundaries which are nothing but contours. We can easily get the coordinates of these contours. This means we can get the locations of the highlighted regions.

Note that there are multiple highlighted regions and each region is encircled by a contour. In our case, the contour having the maximum area is the desired region. Hence, it is better to have as few contours as possible.

In the image above, there are still some unnecessary fragments of the white region. There is still scope of improvement. The idea is to merge the nearby white regions to have fewer contours and for that, we can use another technique known as image dilation.

Image Dilation

This is a convolution operation on an image wherein a kernel (a matrix) is passed over the entire image. Just to give you intuition, the image on the right is the dilated version of the image on the left:

CONCLUSION

The designed algorithm was effectively able to detect the type of violation specified on this project which are denying traffic signal. The convergence of detection for the traffic violation mentioned is dissimilar, since it has a different threshold condition. In order to detect circles in images, you'll need to make use of the cv2.HoughCircles function. It's definitely not the easiest function to use, but with a little explanation, I think you'll get the hang of it. The system provides detection for traffic signal violation.

FUTURE SCOPE

The proposed system can be used for the real time detection of vehicles on the traffic signal, monitoring the vehicles entering and Exit the city or in specific area, parking area of mall, big Organization etc. It helps to Traffic management, transport management to take the decision Capture the photo/image of Vehicle registration plates and store the Data in Database.