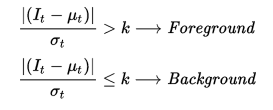
Moving Pedestrian and Vehicle Detection and Tracking

Introduction

Object detection

There are multiple techniques for object detection. For examples, techniques like

Contour detection which are learning based and can detect specific types of object in a given image. However, moving object detection is different since the object we wish to detect can be of any type. To achieve this purpose, it was decided to use the technique of background subtraction from a set of frames to isolate moving objects in a video.

There are many variants of background subtraction, but they all work by comparing each pixel in the current frame with its value in the previous frames. For example, the class BackgroundSubtractorMOG2 in openCV is a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. It computes the running mean and variance of the gaussian distribution function of each pixel’s value over the detection window. At the end of the detection window, if the pixel’s value is outside a confidence interval of each pixel’s mean, then it is a foreground pixel. Or else, it is a background pixel.

The output of the background subtractor is a binary image where the foreground pixels are white pixels and background pixels are dark.

Original frame:

After background subtraction:

It is important to note that background subtraction requires a set of frames to differentiate foreground from background objects. In our experiments, a frame length of 4 was chosen to consistently detect moving objects. This number is purely heuristic and could vary depending on the type of video.

The result of background subtraction is a set of pixels and typically very grainy because of subtle variations in lighting. In order to remove the noise, the following three operations are performed

1. thresholding

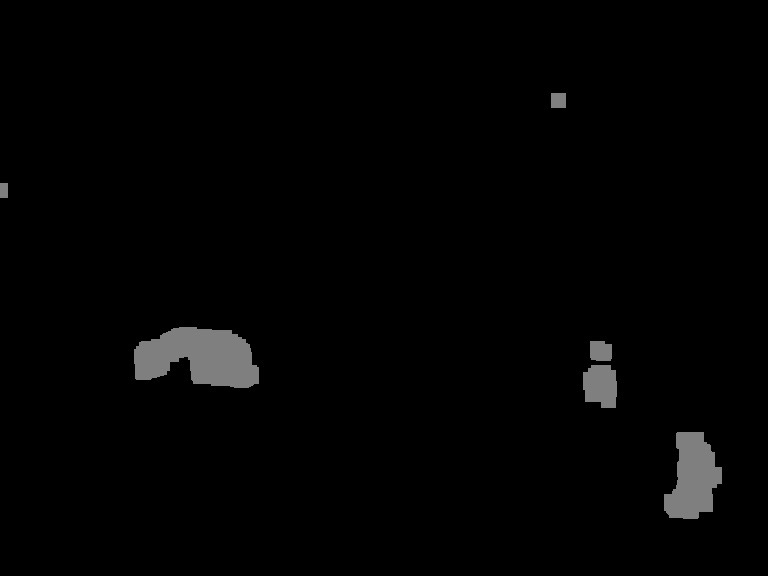
This is a process which removes any pixels which are lower than a threshold value from the foreground to the background.

2. Eroding

This is used to remove any objects which are smaller than a specified kernel with a given size

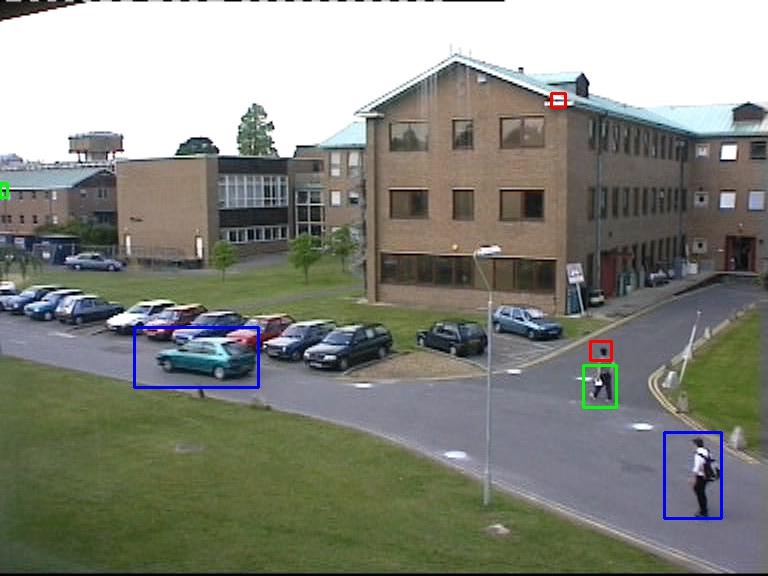
3. Dilation

The remaining objects after erosion are dilated to group the detected pixels better.

The following is the image after these 3 steps are performed.

As it can be seen, the moving objects are clearly delineated in the above image. What is left is the get bounding boxes of the moving objects to be tracked.

In order to detect connected pixels in the above image, contour detection is employed. This groups together related pixels to give a vector of points representing an object. The contour can then be used to compute the smallest encompassing bounding box around it to give us the list of objects to be detected.



As it can be seen above, a lot of post processing is needed to be employed in order to accurately capture each individual object. If the background objects are grainy as it was for one of the people in the image, the bounding rectangles end up being less accurate.

Tracking

At this point, we have the list of objects which are a representation of moving objects and need to be tracked. OpenCV tracker class contains native implementations of many state of the art tracking algorithms like KCF, Median Flow, TLD, MIL and Boosting. From experiments, it was decided to go with the KCF tracker which stands for “kernelized correlation filter”. It uses algorithms based on mean-shift filters to identify the new location of the objects in the current frame based on the previous frame.

It should be noted that tracking can be performed by detecting in each frame. Even though it is bound to be more accurate than tracking algorithms, it is extremely computationally expensive and was not employed in this work.

Areas of improvement

When it comes to object detection, the main problem is the accuracy of the contours if the moving objects are too small or if they blend a lot with the background. In such case, the contours get split into multiple smaller contours. The other issues arises if there are a lot of small moving objects (ex: moving leaves in a tree). This can however be overcome by filtering objects which are smaller than a threshold.

In our implementation, the objects are detected in advance and the tracker takes charge after that. This causes multiple problems. Firstly, if new objects enter into the frame after the detection phase, they are not tracked because they were not detected in the first place. Secondly, if the objects shrink or expand in size as they move, the bounding rectangles do not adjust in size to reflect that change.

In addition, some native issues with the tracker were also observed. Any obstructions in the tracked object cause the tracker to lose track. If the object being tracked blends with the background, this also causes it to lose track. If two tracked objects intersect with each other, the tracker seems to get confused and lose one of the two previously tracked objects.

A very practical solution to these problems is to redo the detection after every few frames in the video and that is a topic reserved for future development.