R219420M Godfrey T kandeya
R219417Y Gift Muzamana

1. Introduction

This report provides a detailed evaluation of the car detection and tracking algorithm that has been implemented in our application using ROI technology. This implemented algorithm is written in Python code, uses the OpenCV library to perform background subtraction, and uses the Euclidean Distance Tracker to count how many cars are moving in the frame (in a certain direction at a certain time). The evaluation is done in terms of the accuracy of the solution of detection, speed, and robustness of occlusions.

2. Solution Description:

The algorithm is based on a mixture of methods for car motion detection and tracking. It uses a region of interest technology to keep track of some parts of the video frame the cars are expected to traverse. OpenCV is the library used for video capture, processing, and visualization. The region of interest is subjected to the background subtraction technique of detection of foreground objects, and a contour analysis is performed to filter and identify the candidates of cars. Detected cars in an image from an input video are ultimately passed to the Euclidean Distance Tracker to follow the cars over successive frames and provide motion estimation.

Background Subtraction Technique:

Our system uses OpenCV's cv2.createBackgroundSubtractorMOG2() function. It creates an object of the Gaussian Mixture-based Background/Foreground Segmentation algorithm, where each pixel of the background is modeled by a mixture of Gaussian distributions to represent the statistical properties of the background.

The algorithm applies cv2.createBackgroundSubtractorMOG2() to the ROI (region of interest) and gets into the video frame using the method object_detector.apply(ROI). The background model is subtracted from the current frame, and finally, a binary mask is achieved. The pixel values for moving cars in the foreground are set to 255 in the binary mask and 0 for the background.

Once the grayscale mask has been binarized using a threshold, the obtained binary image is enhanced. The grayscale mask is intended to be converted into a binary image in which pixe[1]l values are mostly either 0 or 255. The algorithm bins the objects at the threshold value within the foreground from the background more precisely, usually experimentally or adaptively.

Then the binary image is just an extracted foreground, as explained above, and from this binary image, contours are extracted by calling the function cv2.findContours(). In simple terms, contours can be defined as the boundary of connected components in a binary image, which in our case will mean the detected foreground objects (cars). Then the area of these contours is processed to remove the small, spurious contours. Obviously, when the area is greater than some given threshold (say, 5000 pixels), this is treated as a possible candidate for a car.

[1] Car Detection System Using Intelligent Signal Processing | godfrey Kandeya | Gift muzamana

R219420M Godfrey T kandeya
R219417Y Gift Muzamana

Now take the rest of the contours and draw bounding rectangles using cv2.boundingRect() around them. These bounding rectangles should represent the car candidates and are going to track the objects.

Subtraction of the background in the region of interest makes the algorithm detect the moving objects, in this case, the cars, from the background. Following these, contour extraction, filtering, and bounding rectangle estimation steps are made for car detection and successful tracking.

3. **Detection of Cars by the YOLO Library**

Even though our application did not use the YOLO library for car detection, it definitely could benefit from the use of the YOLO algorithm. The algorithm will have the ability to do real-time object detection and high accuracy in the detection of cars. With that, by replacing the background subtraction and contour analysis with the YOLO object detection framework, performance in detection would improve, as it would most likely be.

4. Assessment Criteria:

The performance of this algorithm in detecting and tracking cars was investigated with respect to the following criteria:
a) Accuracy of Detection:

Robustness to Occlusion:

The ability to handle properly car occlusions caused by other objects or other cars was tested. The algorithm's capability to properly treat such an occlusion and robustness in the detection and tracking of cars in difficult conditions were tested.

Performance Evaluation Results:
According to the research analysis, the performance attributes of the car detection and tracking algorithm are:

a) Detection accuracy:
It yielded above-average detection accuracy. The results were also validated using the ground truth annotations, where the results assured the fact that the algorithm was effective in detecting the cars within the ROI.

b) Speed:
The processing speed of the algorithm was found to be satisfactory and could meet real-time requirements for the specified application. On average, the time taken for the processing of each frame was measured, and the values lay within the normal limits.

R219420M Godfrey T kandeya
R219417Y Gift Muzamana

c) Robustness to Occlusions
The algorithm was quite robust against most of the occlusions—it reported excellent performance every time cars were partially occluded by some other object or another car. The algorithm showed effective car detection and tracking even under occlusion, therefore indicating its strength against difficult real-world conditions.

6. Conclusion Conclusively, the ROI approach allows one to come up with an algorithm for car detection and tracking that has shown high performance against the different assessment parameters. This has shown an above-average level of accuracy in detection, as the precision, recall, and F1-score values are at satisfactory levels. Also, the speed of the algorithm is rather good, since in practice it has fulfilled the real-time conditions of the specified application. However, our application worked, but the implementation of the YOLO Library, will enhance the car detection and tracking algorithm toward getting real-time accurate and efficient results for real-world applications.

The table of Task 2 Outcome:

	Total Number of Cars	Cars Per Minute
Julius Nyerere (<—)	56	31
Robert Mugabe (—>)	69	15

- key

< --- Direction of traffic flow (heading left)

< ---- Direction of traffic flow (heading right)

8. References:

https://www.youtube.com/watch?v=nRt2LPRz704

https://www.youtube.com/watch?v=rPGfY-QODh8

https:[2]//www.geeksforgeeks.org/opencv-python-program-vehicle-detection-video-frame/

[2] Car Detection System Using Intelligent Signal Processing | godfrey Kandeya | Gift muzamana