

CM 3065 – Intelligent Signal Processing Final Project Report

Exercise 1 Report

In Exercise 1, I was tasked to develop 2 motion detection applications. For task 1 (Exercise 1.1) of this exercise, I was instructed to make an app for detecting and tracking moving cars from a camera recording using Python and the OpenCV library.

To complete exercise 1, it is required to use background subtraction and frame differencing techniques. Background subtraction separates the moving foreground from the static background. This helps detect objects that are moving in a video and objects that stay still, which is considered the background.

The common background subtraction techniques for motion detection are MOG2 and KNN. The MOG2 method uses K Gaussian Mixture to adapt to changing background. It provides good adaptability to dynamic backgrounds and lighting changes, resulting in high accuracy motion detection. Meanwhile, KNN classifies pixels as background or foreground based on the pixels' feature vector. It is faster and detects every movement effortlessly. However, the drawback for KNN is that it is not as accurate. For task 1, I chose to use the KNN method because it returned smoother results compared to MOG2.

For the frame differencing section of this exercise, I did contour filtering. Contour filtering is a processing technique that filters out unwanted image or colours in Computer Vision. In the case of this task, it is used to exclude small objects because the tasks require me to track only the moving cars.

Task 1 requires the application to detect motion of cars on the main street. This means that it should only detect cars in the lower half of the video. I created a line to specify which is the main street and put some boundaries for where movement should be detected by the app. Therefore, all the cars on the upper half of the video won't be detected by the app.



Figure 1 Task 1 Screenshot

Task 2 (Exercise 1.2) is similar to task 1. It uses similar background subtraction and frame differencing techniques. But some additional requirements must be met which were counting the total number of

cars going to the city centre and the cars per minute count which are going to the city centre. The cars going to the city centre are going to the left side of the screen and off the frame. Since this task requires more counting and accuracy, the MOG2 method is implemented.

To determine which cars are going to the city centre, some adjustments were made to the motion detection boundaries. I minimized the boundaries to only detect motion in the little left area of the screen and added a variable carX to help identify the direction of the cars going to the left. The boundaries for this section had to be very specific to avoid errors. A car counter was also included to count how many cars were detected in the boundaries.

Finally, the cars passed and cars per minute are printed out once the video ends.

```
# Calculations for total cars passed and cars passed per minute
total_time = 177
total_time_minutes = total_time/60
# Calculate cars per minute
carsPM = carsPass / total_time_minutes

print("Total Cars Passed:", carsPass)
print("Video length:", total_time, 's')
print("Cars Per Minute (carsPM):", carsPM)
```

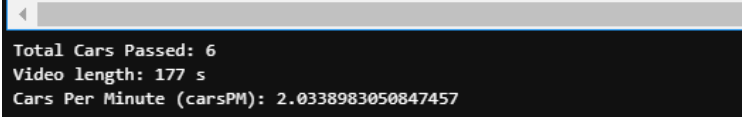


Figure 2 Printing Cars passed to City centre and Cars per minute

The Task 2 table below shows the data generated by the application.

Video title	Total number of cars passing	Video length	Cars Per Minute
Traffic_Laramie_1.mp4	6	177s	2
Traffic_Laramie_2.mp4	4	105s	2