Vehicle Tracking

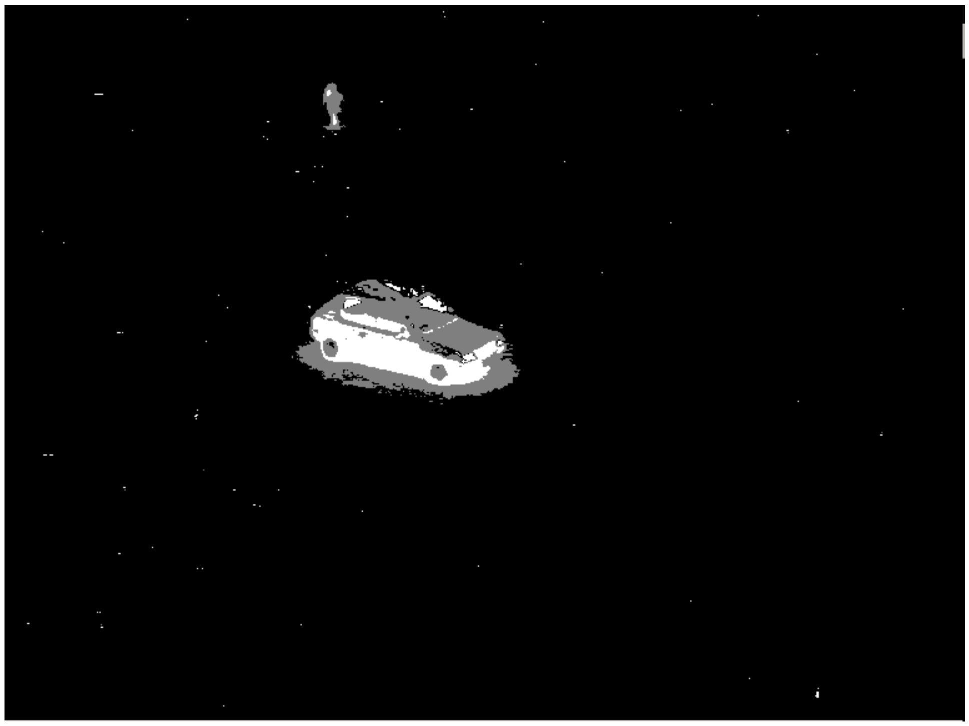
Moaz175888

Project Workflow

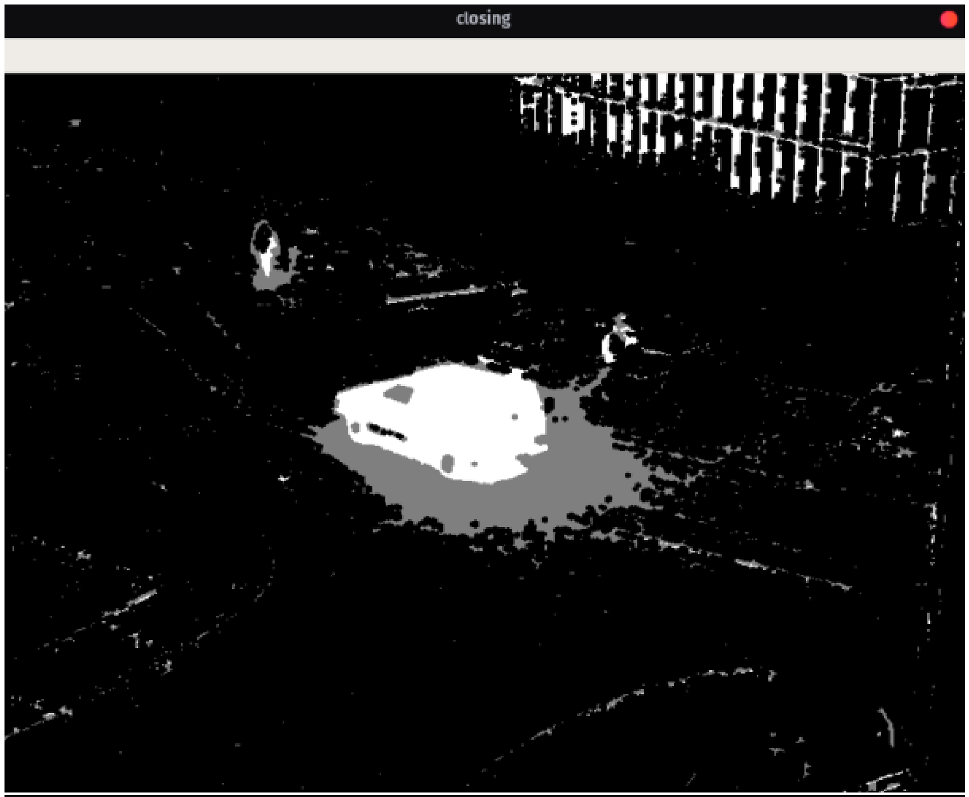
- The program starts by loading the video file, and starts processing the file frame by frame

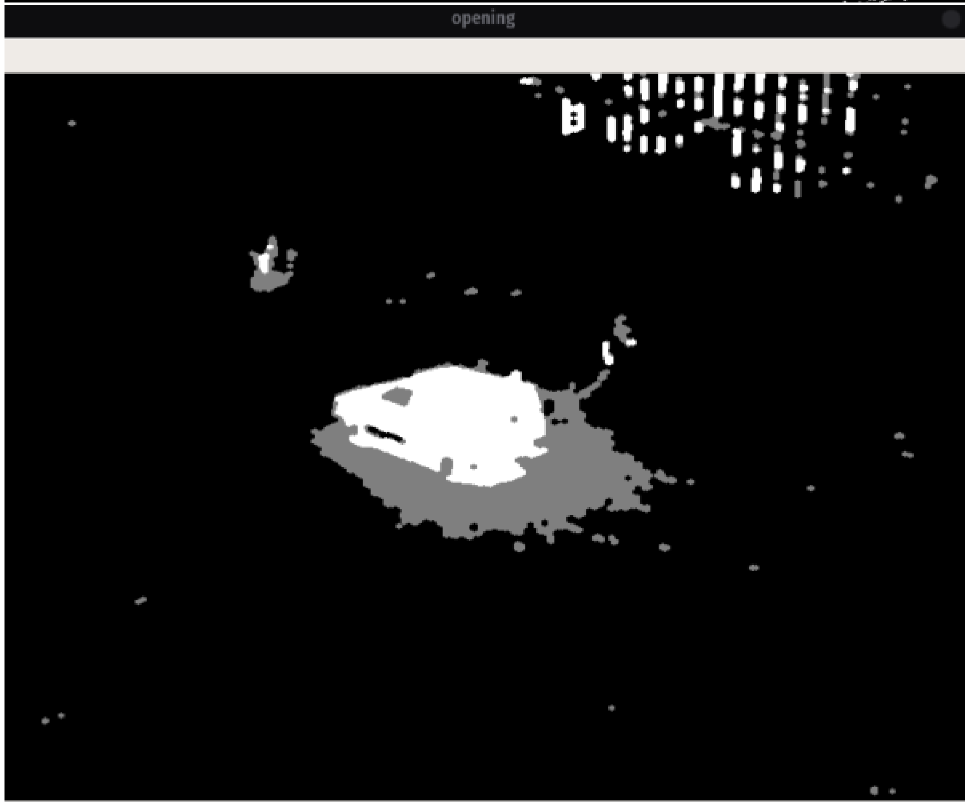
  for each frame:

  - The frame is turned into **grayscale**
  - After that, a **background subtraction** is applied to the gray image
    - It helps to differentiate the moving objects(vehicles) from the road, buildings and still objects.



  - After that, **morphology** is applied to the frame using an **Ellipse kernel**
    - Which makes the objects in the image clearer.





- Then the program applies **thresholding** to the image which makes the cars look more promising and **reduces the noise and removes shadows**

- After that the image is **dilated** to fix the **fragmentation**

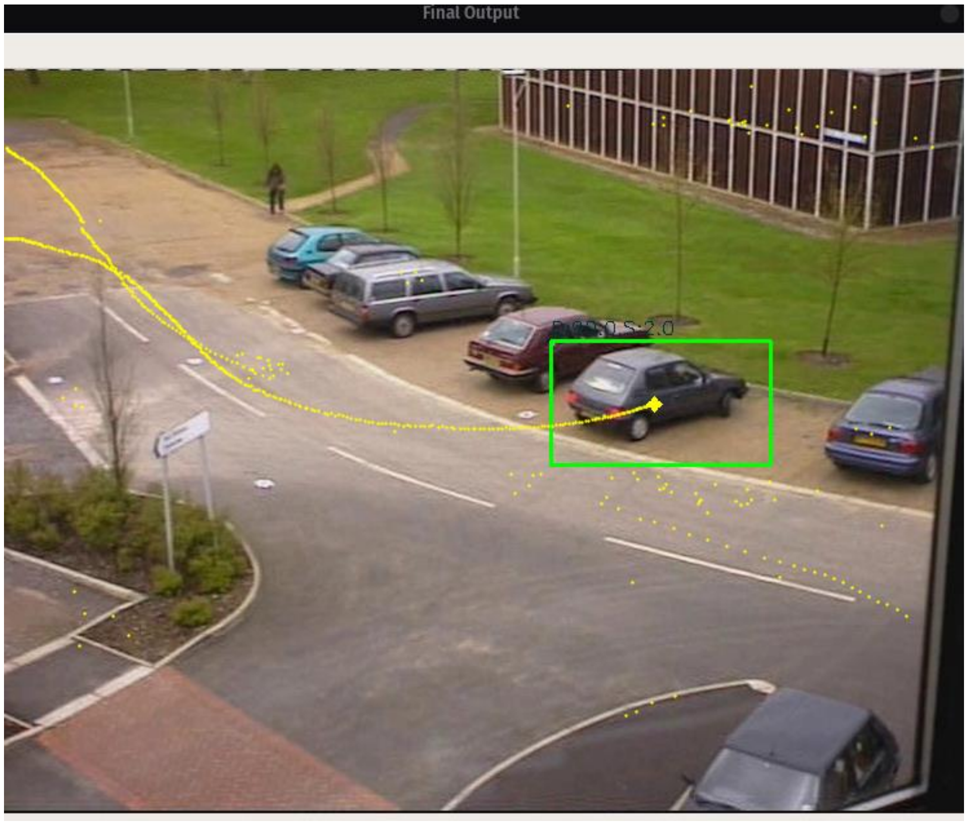
- And finally, the program tries to find the **contours** in the image, and loops through them
    - First, it checks that it only **counts the parent(outer)** contours to avoid having contours inside each other and the need to use algorithms like non-maximum suppression

- Second, it checks the area of the contours if it's between the **min and max areas** given to try to avoid tracking objects that are smaller or bigger than vehicles
- After these two steps, we now have the car contours that we need.
  - First, we calculate the **centroid** of each contour as we will be using it to draw the tracking line and calculating the speed of the vehicle
  - **Speed**: is calculated by saving the centroid location of the vehicle in the previous frame and using it with the current centroid location to calculate how many pixels the vehicle moved per frame

$$d(P,Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

  - And add all the centroids of all the vehicles to the **points** list so that we can draw the tracking lines
  - The angle of the vehicle is also calculated using **fitEllipse**(contour) function
  - And finally, all bounding boxes, centroid, speed, angle of each contour is plotted on the image

- And the tracking points are all drawn



Difficulties and problems

1. **Tracking still vehicles**
    - It's hard to track still vehicles as they are very different and various, we can't use specific algorithm like **color filer**, or it will only pick like **bright colored cars and skip dark cars**
    - The only solution that I managed to make it work was using **Haar cascade** and a pretrained features for cars. Example in '**haarcascade.py**'
2. **Calculating the speed of multiple vehicles in the same time**
    - The first problem that faced me here was that the recognition part of the system needs to be nearly perfect to try to calculate the speed
    - Also having **different videos with different camera location** made me not able to draw like 2 lines on the road with predetermined distance and use them to calculate the speed of the vehicles as the roads will change.
    - **A proposed solution**
        - The solution I thought about is to tracking each separate vehicle using some kind of algorithm like **Euclidian distance** to distinguish between bounding boxes for each vehicle and save their location in a dictionary like datatype and calculate the pixels that each vehicle moved based on the dictionary and the current centroid of each vehicle.
3. Also **calculating the angle of the vehicle was vague** as I didn't know if we should use something like the coordinate system or the angles to the same or to the street.