INTERNET OF THINGS

IBM Naan Mudhalvan Phase-3

Project Tittle: TRAFFIC MANAGEMENT

Configuring IoT devices to measure traffic level and develop a Python script to send collected data to a data-sharing platform involves several steps. Here's a high-level overview of the process:

Build a Vehicle Detection System using OpenCV and Python:

We are all set to build our vehicle detection system! We will be using the computer vision library OpenCV (version – 4.0.0) a lot in this implementation. Let's first import the required libraries and the modules.

Import Libraries

```
import os

import re

import cv2 # opencv library

import numpy as np

  from os.path import isfile, join

  import matplotlib.pyplot as plt
```

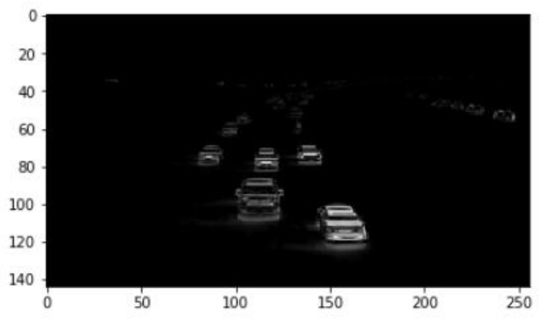
Import Video Frames and Data Exploration:

Keep the frames in a folder named "frames" inside your working directory. From that folder, we will import the frames and keep them in a list and then for data exploration let's display two consecutive frames:

It is hard to find any difference in these two frames, isn't it? As discussed earlier, taking the difference of the pixel values of two consecutive frames will help us observe the moving objects. So, let's use the technique on the above two frames:

```
# convert the frames to grayscale

        grayA = cv2.cvtColor(col_images[i], cv2.COLOR_BGR2GRAY)

        grayB = cv2.cvtColor(col_images[i+1], cv2.COLOR_BGR2GRAY)
```

```
# plot the image after frame differencing

plt.imshow(cv2.absdiff(grayB, grayA), cmap = 'gray')

plt.show()
```

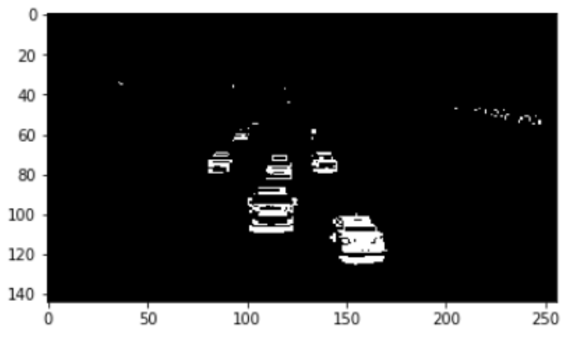


Now we can clearly see the moving objects in the 13th and 14th frames. Everything else that was not moving has been subtracted out.

Image Pre-processing:

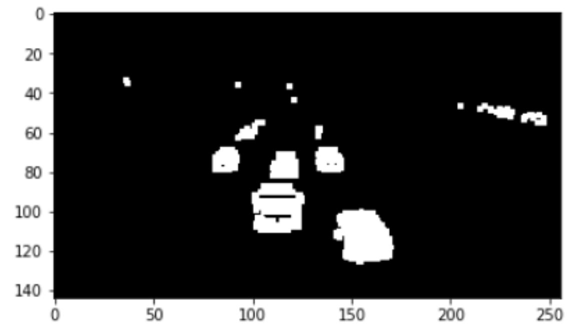
Let’s see what happens after applying thresholding to the above image:

```
diff_image = cv2.absdiff(grayB, grayA)

    # perform image thresholding

    ret, thresh = cv2.threshold(diff_image, 30, 255, cv2.THRESH_BINARY)

    # plot image after thresholding

    plt.imshow(thresh, cmap = 'gray')

    plt.show()
```



Now, the moving objects (vehicles) look more promising and most of the noise (undesired white regions) are gone. However, the highlighted regions are a bit fragmented. So, we can apply image dilation over this image:

```
# apply image dilation
kernel = np.ones((3,3),np.uint8)
dilated = cv2.dilate(thresh,kernel,iterations = 1)


# plot dilated image
plt.imshow(dilated, cmap = 'gray')
plt.show()
```

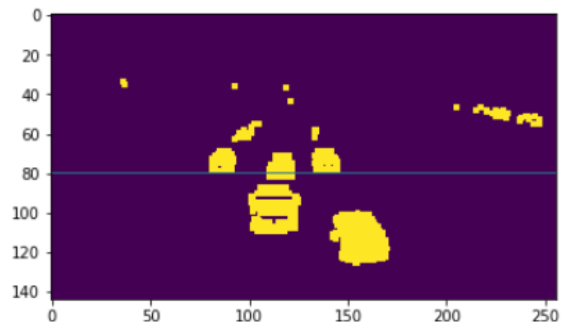


The moving objects have more solid highlighted regions. Hopefully, the number of contours for every object in the frame will not be more than three.

However, we are not going to use the entire frame to detect moving vehicles. We will first select a zone, and if a vehicle moves into that zone, then only it will be detected.

So, let me show you the zone that we will be working with:

```
# plot vehicle detection zone
plt.imshow(dilated)
cv2.line(dilated, (0, 80),(256,80),(100, 0, 0))
plt.show()
```



The area below the horizontal line y = 80 is our vehicle detection zone. We will detect any movement that happens in this zone only. You can create your own detection zone if you want to play around with the concept.

Now let's find the contours in the detection zone of the above frame:

```
# find contours

contours, hierarchy =
cv2.findContours(thresh.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
```

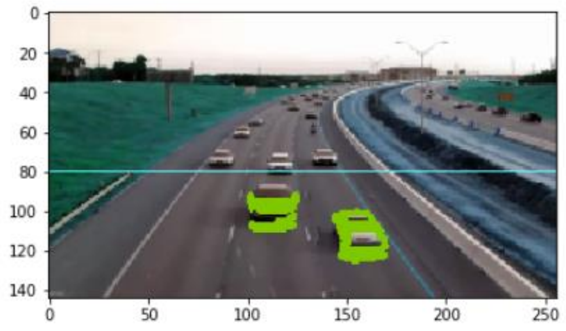
The code above finds all the contours in the entire image and keeps them in the variable 'contours'. Since we have to find only those contours that are present in the detection zone, we will apply a couple of checks on the discovered contours.

The first check is whether the top-left y-coordinate of the contour should be >= 80 (I am including one more check, x-coordinate <= 200). The other check is that the area of the contour should be >= 25. You can find the contour area with the help of the cv2.contourArea( ) function.

```
valid_cntrs = []

for i,cntr in enumerate(contours):
   x,y,w,h = cv2.boundingRect(cntr)
   if (x <= 200) & (y >= 80) & (cv2.contourArea(cntr) >= 25):
      valid_cntrs.append(cntr)


# count of discovered contours
len(valid_cntrs)
```

Next, let's plot the contours along with the original frame:

```
dmy = col_images[13].copy()
cv2.drawContours(dmy, valid_cntrs, -1, (127,200,0), 2)
cv2.line(dmy, (0, 80),(256,80),(100, 255, 255))
plt.imshow(dmy)
plt.show()
```



Cool! Contours of only those vehicles that are inside the detection zone are visible. This is how we will detect vehicles in all the frames.

Vehicle Detection in Videos:

It's time to apply the same image transformations and pre-processing operations on all the frames and find the desired contours. Just to reiterate, we will follow the below steps:

1. Apply frame differencing on every pair of consecutive frames
2. Apply image thresholding on the output image of the previous step
3. Perform image dilation on the output image of the previous step
4. Find contours in the output image of the previous step
5. Shortlist contours appearing in the detection zone
6. Save frames along with the final contours

```
# kernel for image dilation

kernel = np.ones((4,4),np.uint8)


# font style
font = cv2.FONT_HERSHEY_SIMPLEX
```

```
# directory to save the ouput frames
pathIn = "contour_frames_3/"


for i in range(len(col_images)-1):

    # frame differencing
    grayA = cv2.cvtColor(col_images[i], cv2.COLOR_BGR2GRAY)
    grayB = cv2.cvtColor(col_images[i+1], cv2.COLOR_BGR2GRAY)
    diff_image = cv2.absdiff(grayB, grayA)

    # image thresholding
    ret, thresh = cv2.threshold(diff_image, 30, 255, cv2.THRESH_BINARY)

    # image dilation
    dilated = cv2.dilate(thresh,kernel,iterations = 1)

    # find contours
    contours, hierarchy = cv2.findContours(dilated.copy(),
cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)

    # shortlist contours appearing in the detection zone
    valid_cntrs = []
    for cntr in contours:
        x,y,w,h = cv2.boundingRect(cntr)
        if (x <= 200) & (y >= 80) & (cv2.contourArea(cntr) >= 25):
            if (y >= 90) & (cv2.contourArea(cntr) < 40):
                break
            valid_cntrs.append(cntr)

    # add contours to original frames
    dmy = col_images[i].copy()
    cv2.drawContours(dmy, valid_cntrs, -1, (127,200,0), 2)

    cv2.putText(dmy, "vehicles detected: " + str(len(valid_cntrs)), (55, 15), font, 0.6, (0,
180, 0), 2)
    cv2.line(dmy, (0, 80),(256,80),(100, 255, 255))
    cv2.imwrite(pathIn+str(i)+'.png',dmy)
```

Video Preparation:

Here, we have added contours for all the moving vehicles in all the frames. It's time to stack up the frames and create a video:

```
# specify video name
pathOut = 'vehicle_detection_v3.mp4'


# specify frames per second
fps = 14.0
```

Next, we will read the final frames in a list:

```
frame_array = []
files = [f for f in os.listdir(pathIn) if isfile(join(pathIn, f))]
    files.sort(key=lambda f: int(re.sub('\D', '', f)))


    for i in range(len(files)):
        filename=pathIn + files[i]

        #read frames
        img = cv2.imread(filename)
        height, width, layers = img.shape
        size = (width,height)

        #inserting the frames into an image array
        frame_array.append(img)
```

Finally, we will use the below code to make the object detection video:

```
out = cv2.VideoWriter(pathOut,cv2.VideoWriter_fourcc(*'DIVX'), fps, size)


for i in range(len(frame_array)):
    # writing to a image array
    out.write(frame_array[i])


out.release()
```

Team Members:

ARASU.C (810721106001 , arasu.c@care.ac.in**)**

SAMRUTH SRIRAM.D (810721106016, samruthsriram.d@care.ac.in**)**

HANISH.K.A (810721106007, hanish.ka@care.ac.in**)**

SIVAGANAPATHY.R (810721106018, sivaganapathy.r@care.ac.in**)**