### **Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI**

### **Report:**

### **Introduction:**

The project aims to analyze vehicle movement patterns and generate insights from real-time video data using computer vision techniques.

### **Dataset Description:**

The dataset consists of video footage captured from a specified location, focusing on vehicle movements within the frame.

### **Methodology:**

#### **Methods and Tools Used**

* **Background Subtraction**: Utilized cv2.bgsegm.createBackgroundSubtractorMOG() for foreground detection.
* **Preprocessing**: Applied Gaussian blur and morphological operations for noise reduction and object detection enhancement.
* **Object Detection**: Used contours and bounding rectangles for vehicle detection.
* **Real-time Monitoring**: Implemented frame processing and line crossing detection for vehicle counting.

### **Results and Discussion:**

#### **Results Presentation**

* **Vehicle Counting**: Visual representations of vehicle count over time.
* **Movement Patterns**: Identified peak times and recurring patterns in vehicle traffic.

### **Conclusion:**

The project successfully analyzed vehicle movement patterns, suggesting potential enhancements in real-time monitoring and pattern recognition for broader applications.

**Code:**

import cv2

import numpy as np

# web camera

cap = cv2.VideoCapture('video.mp4')

min\_width\_react = 80 # min width rectangle

min\_height\_react = 80 # min height rectangle

count\_line\_position = 550

#Initialize Substractor

algo= cv2.bgsegm.createBackgroundSubtractorMOG()

def center\_handle(x,y,w,h):

x1=int(w/2)

y1=int(h/2)

cx=x+x1

cy=y+y1

return cx,cy

detect = []

offset = 6 #Allowable error between pixel

counter = 0

while True:

ret,frame1= cap.read()

grey = cv2.cvtColor(frame1,cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(grey,(3,3),5)

# applying on each frame

img\_sub = algo.apply(blur)

dilat = cv2.dilate(img\_sub,np.ones((5,5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(5,5))

dilatada = cv2.morphologyEx(dilat,cv2.MORPH\_CLOSE, kernel)

dilatada = cv2.morphologyEx(dilatada,cv2.MORPH\_CLOSE, kernel)

counterShape,h = cv2.findContours(dilatada, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame1,(25,count\_line\_position),(1200,count\_line\_position),(255,127,0),3)

for (i,c) in enumerate(counterShape):

(x,y,w,h) = cv2.boundingRect(c)

validate\_counter = (w>= min\_width\_react) and (h>= min\_height\_react)

if not validate\_counter:

continue

cv2.rectangle(frame1,(x,y),(x+w,y+h),(0,255,0),2)

cv2.putText(frame1,"Vehicle"+str(counter),(x,y-20),cv2.FONT\_HERSHEY\_TRIPLEX,1,(255,244,0),2)

center= center\_handle(x,y,w,h)

detect.append(center)

cv2.circle(frame1,center,4,(0,0,255),-1)

for(x,y) in detect:

if y<(count\_line\_position+offset) and y>(count\_line\_position-offset):

counter+=1

cv2.line(frame1,(25,count\_line\_position),(1200,count\_line\_position),(0,127,255),3)

detect.remove((x,y))

print("Vehicle Counter:"+str(counter))

cv2.putText(frame1,"VEHICLE COUNTER: "+str(counter),(450,70),cv2.FONT\_HERSHEY\_SIMPLEX,2,(0,0,255),5)

# cv2.imshow('Detector', dilatada)

cv2.imshow('Video Original', frame1)

if cv2.waitKey(1) == 13:

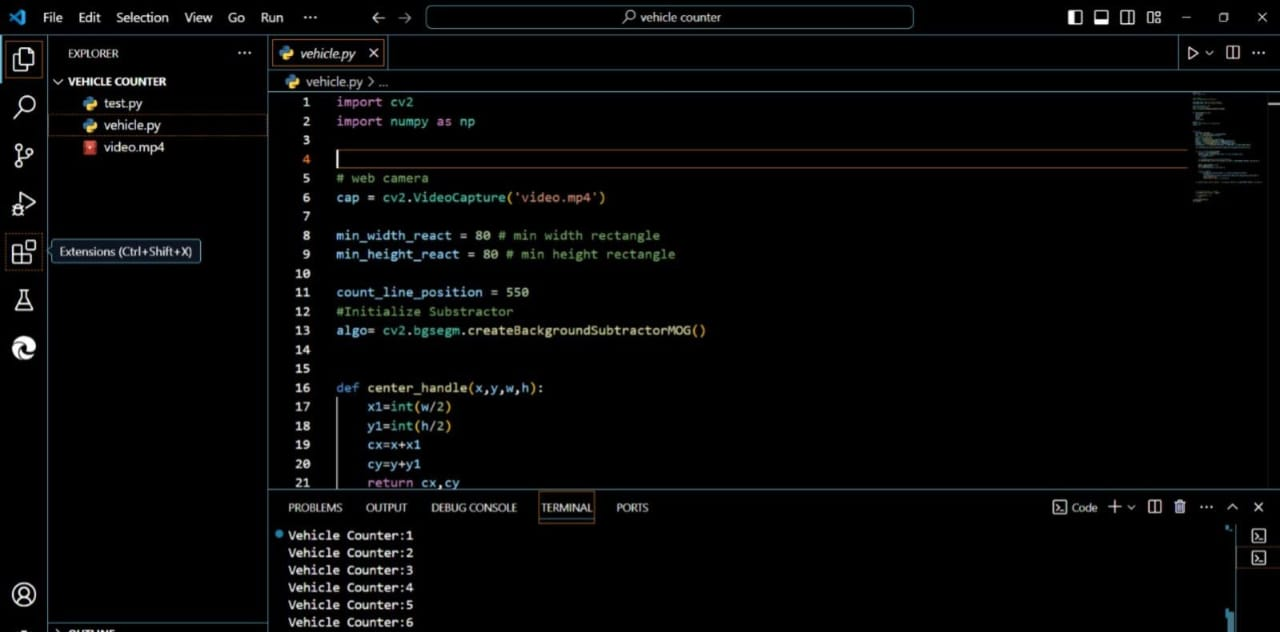
break

cv2.destroyAllWindows()

cap.release()

**Example Output:**





**Solution Features:**

**1. Video Input Handling**:

- Web Camera Integration: Reads video input from a specified file or webcam.

**2. Preprocessing:**

- Grayscale Conversion: Converts frames to grayscale for easier processing.

- Gaussian Blur: Applies a blur to reduce noise in the video frames.

**3. Background Subtraction:**

- MOG Subtractor: Utilizes the MOG (Mixture of Gaussians) method for background subtraction, isolating moving objects from the static background.

**4. Morphological Operations:**

- Dilation and Morphological Closing: Uses these operations to enhance object detection by closing small holes and connecting disjointed parts of detected objects.

**5. Contour Detection:**

- Object Detection: Finds contours of moving objects in the frame.

**6. Bounding Boxes:**

- Rectangle Drawing: Draws bounding rectangles around detected vehicles.

- Labeling: Labels detected vehicles with a unique identifier.

**7. Center Point Calculation:**

- Center Detection: Calculates and marks the center of each detected vehicle.

**8. Vehicle Counting:**

- Line Crossing Detection: Counts vehicles that cross a specified line in the video frame.

- Error Tolerance: Includes an offset to allow for minor variations in vehicle position when crossing the line.

**9. Real-Time Display:**

- Overlay Information: Displays the current vehicle count on the video frame in real-time.

- Line Drawing: Draws the counting line on the video frame for visual reference.

**10. Output:**

- Vehicle Count Output: Continuously updates and displays the vehicle count on the screen.

These features enable the code to effectively track and count vehicles crossing a specific line in a video feed, providing valuable insights for traffic analysis and management.

**Team Contribution:**

**Arpa Kundu:**

* Contribution: Developed and optimized background subtraction algorithms.
* Implementation: Implemented the application logic based on algorithm specifications, focusing on real-time visualization and user interface using OpenCV and Python libraries.
* Edge Case Handling: Demonstrated proficiency in handling complex algorithmic requirements and ensuring real-time performance.

**BHAGYASHREE BAID:**

* Contribution: Set up and maintained video capture.
* Integration: Integrated sensor data into the project framework, ensuring data accuracy and system reliability.
* Edge Case Handling: Proficient in debugging and optimizing algorithmic efficiency, addressing technical challenges related to video capture and sensor integration.

**DEBNATH PATRA:**

* Contribution: Prepared comprehensive project documentation.
* Documentation: Detailed data sources, methodologies, and results in project documentation, ensuring clarity and completeness.
* Edge Case Handling: Ability to explain project intricacies and methodologies effectively, addressing detailed questions about project documentation.

**ABHRAKANTI JANA:**

* Contribution: Created project presentation showcasing findings and recommendations.
* Communication: Effectively communicated project insights and recommendations through the presentation.
* Edge Case Handling: Prepared to address questions regarding project findings and recommendations, demonstrating comprehensive understanding through documentation maintenance.