**BÁO CÁO CÔNG VIỆC**

**Công việc số:** 69

**Mô tả công việc:** Chạy lại và hiểu cách tính diện tích phương tiện trên đường

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 06/01/2025

**Ngày kết thúc:** 10/02/2025

***Ghi chú:***

**NỘI DUNG TÀI LIỆU**

[**1. Mục tiêu 3**](#_x03kzrg5yz6t)

[2. Input 3](#_gkjytu3ekg85)

[3. Output 3](#_17hp5itb6p2y)

[4. Cách giải quyết vấn đề 3](#_acxuj2vj6ezx)

[5. Lệnh chạy file mã nguồn 3](#_31bm04nvploe)

[6. Code tìm thêm 4](#_9ef89hcup7t)

[7. Mã nguồn chi tiết 4](#_z5m0wpkdyo3g)

**Lệnh chạy file mã nguồn (Bổ sung cài đặt nếu cần)**

**Code tìm thêm**

**Công thức**

**Thử nghiệm**

**Nhận xét**

# 1. Mục tiêu

* Xác định diện tích mà một chiếc ô tô chiếm dụng trên lòng đường bằng cách xử lý hình ảnh.

# 2. Input

* Ảnh ban đầu chưa qua xử lý.
* Ảnh đánh dấu phần lòng đường (phần lòng đường màu trắng, phần còn lại màu đen).
* Ảnh đánh dấu phần lòng đường và phần không phải lòng đường.

# 3. Output

* Diện tích lòng đường mà ô tô chiếm dụng (tính bằng pixel).
* Hình ảnh sau khi nhận diện các phương tiện và diện tích chiếm dụng của chúng.
* Số lượng phương tiện trên đường.
* Phần trăm diện tích lòng đường bị chiếm dụng.

# 4. Cách giải quyết vấn đề

1. **Tiền xử lý ảnh:**
   * Đọc ảnh đầu vào gồm ảnh gốc và ảnh mặt nạ.
   * Sử dụng phép toán bitwise\_and để giữ lại phần lòng đường có phương tiện chiếm dụng.
2. **Xác định diện tích tổng của lòng đường:**
   * Chuyển ảnh mặt nạ về ảnh xám.
   * Tìm các contours để xác định vùng lòng đường có phương tiện.
   * Tính diện tích contour lớn nhất để lấy tổng diện tích lòng đường trong ảnh.
3. **Nhận diện phương tiện và xác định diện tích chiếm dụng:**
   * Sử dụng YOLOv8 với mô hình yolov8s-seg.pt để phát hiện các phương tiện trong ảnh.
   * Nhận diện các đối tượng thuộc danh sách: car, truck, motorcycle, bicycle, bus, person.
   * Sử dụng np.count\_nonzero để đếm số pixel thuộc về các phương tiện trong ảnh.
4. **Tính toán kết quả:**
   * Tính tổng diện tích chiếm dụng của tất cả phương tiện.
   * Đếm số lượng phương tiện trong ảnh.
   * Tính phần trăm diện tích lòng đường bị chiếm dụng:

# 5. Lệnh chạy file mã nguồn

pip install ultralytics opencv-python numpy matplotlib

python main.py

# 6. Code tìm thêm

* **Đọc ảnh:** cv2.imread()
* **Xử lý ảnh:** cv2.bitwise\_and()
* **Tìm contours:** cv2.findContours()
* **Vẽ contours:** cv2.drawContours()
* **Tính diện tích contours lớn nhất:** cv2.contourArea()
* **Nhận diện phương tiện:** YOLO.predict()
* **Tính diện tích chiếm dụng:** np.count\_nonzero()
* **Tính tổng diện tích chiếm dụng:** Sum\_area += np.count\_nonzero(results[0].masks.data[i])
* **Tính phần trăm lòng đường bị chiếm dụng:** np.round(Sum\_area/area\*100, 2)
* **Hiển thị kết quả:** print() và plt.imshow()

# 7. Mã nguồn chi tiết

| # -\*- coding: utf-8 -\*-  """ghép  Automatically generated by Colab.  Original file is located at  https://colab.research.google.com/drive/12IsLSnECHzrKO7AXWpkF58na9k5QrStO  """  !pip install ultralytics opencv-python  import cv2  import numpy as np  from ultralytics import YOLO  from collections import defaultdict  import time  import matplotlib.pyplot as plt  from google.colab.patches import cv2\_imshow  """# 1. Phần tính diện tích phương tiện chiếm dụng lòng đường"""  def calculate\_road\_occupancy(original\_frame, processed\_frame, model):  # Chuyển ảnh lòng đường thành ảnh nhị phân  gray\_road\_mask = cv2.cvtColor(processed\_frame, cv2.COLOR\_BGR2GRAY)  \_, road\_mask = cv2.threshold(gray\_road\_mask, 200, 255, cv2.THRESH\_BINARY)  road\_area = np.count\_nonzero(road\_mask == 255)  # Nhận diện phương tiện  results = model(original\_frame)  names = model.names  vehicle\_list = ["car", "truck", "motorcycle", "bicycle", "bus", "person"]  sum\_vehicle\_area = 0  num\_vehicles = 0  for i in range(len(results[0].masks)):  class\_id = int(results[0].boxes.cls[i])  if names[class\_id] in vehicle\_list:  mask = results[0].masks.data[i]  vehicle\_area = np.count\_nonzero(mask.cpu().numpy())  # vehicle\_area = np.count\_nonzero(mask.cpu().numpy())  sum\_vehicle\_area += vehicle\_area  num\_vehicles += 1  # Tính phần trăm chiếm dụng  percentage\_occupancy = (sum\_vehicle\_area / road\_area) \* 100 if road\_area > 0 else 0  return {  "road\_area": road\_area,  "num\_vehicles": num\_vehicles,  "sum\_vehicle\_area": sum\_vehicle\_area,  "percentage\_occupancy": percentage\_occupancy  }  # Ví dụ sử dụng  model = YOLO("yolov8s-seg.pt")  original\_frame = cv2.imread("/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job69/img7.jpg")  processed\_frame = cv2.imread("/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job69/img5\_mask2.jpg")  # result = calculate\_road\_occupancy(original\_frame, processed\_frame, model)  # print(result)  """# 2. Phần tính tốc độ"""  class SpeedDetector:  def \_\_init\_\_(self, model\_path='yolov8n.pt', lane\_width\_meters=3.5, skip\_frames=2):  # Khởi tạo mô hình YOLO để nhận diện đối tượng  self.model = YOLO(model\_path)  # Chiều rộng làn đường chuẩn (mặc định 3.5 mét)  self.lane\_width\_meters = lane\_width\_meters  # Từ điển để theo dõi các phương tiện  self.vehicle\_tracking = {}  # Cờ kiểm tra đã hiệu chuẩn chưa  self.calibration\_done = False  # Số pixel trên một mét  self.meters\_per\_pixel = None  # ID tiếp theo cho phương tiện mới  self.next\_id = 0  # Số khung hình bỏ qua giữa các lần xử lý  self.skip\_frames = skip\_frames  # Từ điển ánh xạ các lớp phương tiện được hỗ trợ  self.vehicle\_classes = {  2: 'Car',  3: 'Motorcycle',  5: 'Bus',  7: 'Truck'  }  # Khởi tạo từ điển thống kê phương tiện cho mỗi khung hình  self.frame\_vehicle\_stats = defaultdict(lambda: {  'total\_vehicles': 0,  'vehicle\_counts': {cls: 0 for cls in self.vehicle\_classes.values()},  'speeds': []  })  def calculate\_speed(self, pos1, pos2, time\_diff):  # Kiểm tra đã hiệu chuẩn chưa  if not self.calibration\_done:  raise ValueError("Cần hiệu chuẩn trước khi tính vận tốc!")  # Tránh chia cho số 0  if time\_diff < 0.001:  return 0.0  # Tính khoảng cách giữa hai điểm  dx = pos2[0] - pos1[0]  dy = pos2[1] - pos1[1]  distance\_pixels = np.sqrt(dx\*\*2 + dy\*\*2)  # Chuyển đổi khoảng cách từ pixel sang mét  distance\_meters = distance\_pixels \* self.meters\_per\_pixel  # Tính vận tốc (mét/giây)  speed\_mps = distance\_meters / time\_diff  # Chuyển đổi sang km/h và giới hạn vận tốc  speed\_kmh = min(max(speed\_mps \* 3.6, 0), 200)  return speed\_kmh  def process\_video(self, video\_path, output\_path='output\_video.mp4', lane\_width\_pixels=100):  # Hiệu chuẩn số pixel trên mét nếu chưa thực hiện  if not self.calibration\_done:  self.meters\_per\_pixel = self.lane\_width\_meters / lane\_width\_pixels  self.calibration\_done = True  # Mở video  cap = cv2.VideoCapture(video\_path)  if not cap.isOpened():  raise ValueError("Không thể mở video!")  # Lấy thông số video  fps = cap.get(cv2.CAP\_PROP\_FPS)  frame\_time = 1/fps if fps > 0 else 0.033  width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  # Tính thời gian thực của khung hình  actual\_frame\_time = frame\_time \* (self.skip\_frames + 1)  # Thiết lập ghi video  fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  out = cv2.VideoWriter(output\_path, fourcc, fps/(self.skip\_frames + 1), (width, height))  # Khởi tạo bộ đếm khung hình  frame\_count = 0  processed\_count = 0  # Vòng lặp xử lý video  while cap.isOpened():  ret, frame = cap.read()  if not ret:  break  frame\_count += 1  # Bỏ qua các khung hình không cần thiết  if frame\_count % (self.skip\_frames + 1) != 0:  continue  processed\_count += 1  # Đặt lại thống kê cho khung hình hiện tại  current\_frame\_stats = self.frame\_vehicle\_stats[processed\_count]  current\_frame\_stats['total\_vehicles'] = 0  for cls in self.vehicle\_classes.values():  current\_frame\_stats['vehicle\_counts'][cls] = 0  current\_frame\_stats['speeds'].clear()  # Nhận diện đối tượng  results = self.model(frame, stream=True)  current\_frame\_ids = set()  # Xử lý từng đối tượng được nhận diện  for result in results:  boxes = result.boxes  for box in boxes:  class\_id = int(box.cls[0].cpu().numpy())  if class\_id in self.vehicle\_classes:  vehicle\_class = self.vehicle\_classes[class\_id]  # Tính toán trung tâm phương tiện  x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()  x\_center = (x1 + x2) / 2  y\_center = (y1 + y2) / 2  current\_pos = (x\_center, y\_center)  # Kiểm tra nếu là loại phương tiện được hỗ trợ  # if int(box.cls[0]) in self.vehicle\_classes:  # vehicle\_class = self.vehicle\_classes[int(box.cls[0])]  # # Tính tọa độ trung tâm của phương tiện  # x\_center = (box.xyxy[0][0] + box.xyxy[0][2]) / 2  # y\_center = (box.xyxy[0][1] + box.xyxy[0][3]) / 2  # current\_pos = (x\_center, y\_center)  # Cập nhật thống kê  current\_frame\_stats['total\_vehicles'] += 1  current\_frame\_stats['vehicle\_counts'][vehicle\_class] += 1  # Tìm ID phương tiện phù hợp  min\_dist = float('inf')  matched\_id = None  # Kiểm tra với các phương tiện đã theo dõi trước đó  for obj\_id, data in self.vehicle\_tracking.items():  if data['positions'] and processed\_count - data['last\_update'] < 5:  last\_pos = data['positions'][-1]  dist = np.sqrt((last\_pos[0] - x\_center)\*\*2 +  (last\_pos[1] - y\_center)\*\*2)  if dist < min\_dist and dist < 100:  min\_dist = dist  matched\_id = obj\_id  # Tạo ID mới nếu chưa tìm thấy  if matched\_id is None:  matched\_id = self.next\_id  self.next\_id += 1  self.vehicle\_tracking[matched\_id] = {  'type': vehicle\_class,  'positions': [],  'timestamps': [],  'velocities': [],  'last\_update': processed\_count  }  current\_frame\_ids.add(matched\_id)  track = self.vehicle\_tracking[matched\_id]  # Cập nhật thông tin vị trí và thời gian  track['positions'].append(current\_pos)  track['timestamps'].append(processed\_count \* actual\_frame\_time)  track['last\_update'] = processed\_count  # Tính vận tốc  speed = 0  if len(track['positions']) >= 2:  prev\_pos = track['positions'][-2]  prev\_time = track['timestamps'][-2]  current\_time = track['timestamps'][-1]  time\_diff = current\_time - prev\_time  speed = self.calculate\_speed(prev\_pos, current\_pos, time\_diff)  track['velocities'].append(speed)  current\_frame\_stats['speeds'].append(speed)  # Lấy trung bình vận tốc 3 khung hình gần nhất  if len(track['velocities']) > 3:  speed = np.mean(track['velocities'][-3:])  # Vẽ hình chữ nhật và ghi chú cho phương tiện  x1, y1, x2, y2 = box.xyxy[0]  cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)),  (0, 255, 0), 2)  cv2.putText(frame,  f'{vehicle\_class} ID:{matched\_id} Speed:{speed:.1f} km/h',  (int(x1), int(y1)-10),  cv2.FONT\_HERSHEY\_SIMPLEX,  0.5,  (0, 255, 0),  2)  result = calculate\_road\_occupancy(frame, processed\_frame, model)  print(result)  # Hiển thị thống kê phương tiện trên khung hình  y\_offset += 25  cv2.putText(frame, f"Ty le chiem dung long duong: {result['percentage\_occupancy']:.2f}%",  (10, y\_offset),  cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (255, 0, 0), 2)  y\_offset = 30  cv2.putText(frame, f"Frame {processed\_count}",  (10, y\_offset),  cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)  y\_offset += 30  cv2.putText(frame, f"Total Vehicles: {current\_frame\_stats['total\_vehicles']}",  (10, y\_offset),  cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (255, 0, 0), 2)  y\_offset += 25  for vehicle\_type, count in current\_frame\_stats['vehicle\_counts'].items():  cv2.putText(frame, f"{vehicle\_type}: {count}",  (10, y\_offset),  cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)  y\_offset += 25  # Xóa các phương tiện không còn xuất hiện  for obj\_id in list(self.vehicle\_tracking.keys()):  if (obj\_id not in current\_frame\_ids and  processed\_count - self.vehicle\_tracking[obj\_id]['last\_update'] > 5):  del self.vehicle\_tracking[obj\_id]  # Ghi và hiển thị khung hình  out.write(frame)  cv2\_imshow(frame)  if cv2.waitKey(1) & 0xFF == ord('q'):  break  # Giải phóng tài nguyên  cap.release()  out.release()  cv2.destroyAllWindows()  # Trả về thông tin theo dõi phương tiện và thống kê  return self.vehicle\_tracking, self.frame\_vehicle\_stats  """# 3. Hàm Main"""  def main():  # Tạo instance của SpeedDetector  detector = SpeedDetector(  lane\_width\_meters=3.5, # Chiều rộng làn đường  skip\_frames=3 # Số khung hình bỏ qua giữa các lần xử lý  )  # Đường dẫn video đầu vào  video\_path = '/content/drive/MyDrive/NGANGIANGINTERNSHIP/Job69/b.mp4'  # Xử lý video  detector.process\_video(  video\_path=video\_path,  output\_path='output\_speed.mp4',  lane\_width\_pixels=243 # Chiều rộng làn đường trong ảnh  )  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| --- |