**BÁO CÁO CÔNG VIỆC**

**Công việc số:** 62

**Mô tả công việc:** Tính tốc độ của phương tiện giao thông qua video

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 12/12/2024

**Ngày kết thúc:** \_\_\_/\_\_\_/2024

***Ghi chú:*** *Tạm dừng ngày 16/12/2014, tiếp tục ngày 03/01/2025 (18 ngày) để làm Job 63*

**NỘI DUNG TÀI LIỆU**

[**1. Cách tính vận tốc của phương tiện thông qua các frame 3**](#_ytyue9kzsw5q)

[1.1. Phát hiện đối tượng 3](#_r4jx29p28708)

[1.2. Tính tọa độ tâm của bounding box 3](#_ebk14zazqk5l)

[1.3. Theo dõi đối tượng qua các frame 3](#_clzash8fc2cu)

[1.4. Tính vận tốc trong pixel/frame 3](#_i9nedvc5jshk)

[5. Chuyển đổi vận tốc từ pixel sang đơn vị thực tế 4](#_o8fb7t7iqsoq)

[6. Lọc dữ liệu vận tốc (Tùy chọn) 4](#_t3caj2aslhvu)

[7. Ghi chú và hiển thị vận tốc 4](#_47mlia3u6rho)

[**2. DPI 4**](#_ygghvsiwqumy)

[2.1. DPI của Ảnh 4](#_b8qprlbplp2v)

[2.2. Camera và Độ phân giải 5](#_8sxdt7cel1xa)

[a. Độ phân giải cảm biến (Megapixel) 5](#_7u276aeejd4v)

[b. Kích thước cảm biến 5](#_xi773koi695f)

[c. Ống kính và thông số kỹ thuật 5](#_rdsvtleix1jk)

[d. Tốc độ và tính năng 5](#_lwahlok5miua)

[**3. Chuyển đổi từ Inch sang Mét 6**](#_hrbf94oaa95)

[**4. Sử dụng CSharp để check DPI của ảnh và camera 6**](#_koz54f6r38uo)

[4.1. Kiểm tra DPI của ảnh 6](#_4is83ej2r5g0)

[4.2. Kiểm tra DPI của Camera 7](#_2y1kxki6hiw9)

[**5. Ứng dụng tính vận tốc của phương tiện 9**](#_4himxroeeimb)

[1. Định nghĩa bài toán 9](#_o3nn36kne145)

[2. Thành phần chính trong giải pháp 9](#_cs6dw0skvki5)

[2.1. Phát hiện đối tượng (Object Detection) 9](#_ldlp7yerrw1r)

[2.2. Theo dõi đối tượng (Object Tracking) 9](#_8j5yx15yf8pt)

[2.3. Hiệu chuẩn camera (Calibration) 9](#_8jjz4zu598wn)

[2.4. Tính vận tốc 10](#_jt6hyzk7f64d)

[2.5. Skip Frame (Tăng hiệu suất) 10](#_f320gxhi2zpz)

[2.6. Xuất video đầu ra 10](#_u1wt27chhp1x)

[3. Mã nguồn 10](#_nf69oufohiko)

# 

# **1. Cách tính vận tốc của phương tiện thông qua các frame**

## 1.1. Phát hiện đối tượng

**Mục tiêu:** Phát hiện các phương tiện giao thông trong từng frame của video.

**Phương pháp:** Sử dụng mô hình YOLO để phát hiện bounding box.

**Công thức:** Bounding box có dạng:

* ​: Tọa độ góc trên trái.
* ​: Tọa độ góc dưới phải.

## 1.**2. Tính tọa độ tâm của bounding box**

**Mục tiêu:** Xác định tâm của bounding box để theo dõi chuyển động. Tâm được biểu diễn dưới dạng

Trong đó:

## 1.**3. Theo dõi đối tượng qua các frame**

**Mục tiêu:** Kết nối bounding box của cùng một phương tiện qua các frame.

**Phương pháp:**

* Sử dụng thuật toán theo dõi như SORT, DeepSORT hoặc gán ID theo logic đơn giản.
* Lưu trữ lịch sử tọa độ tâm CtC\_tCt​ của từng đối tượng ttt.

## 1.**4. Tính vận tốc trong pixel/frame**

**Mục tiêu:** Xác định vận tốc dựa trên sự thay đổi vị trí tâm giữa các frame.

Trong đó:

* : Tọa độ tâm ở frame trước
* : Tọa độ tâm ở frame sau
* : Thời gian giữa hai frame

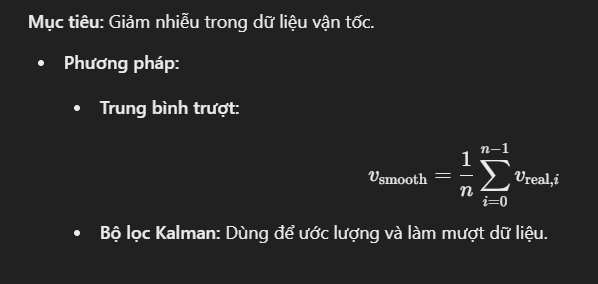
## **5. Chuyển đổi vận tốc từ pixel sang đơn vị thực tế**

**Mục tiêu:** Biến đổi vận tốc từ pixel/frame sang m/s hoặc km/h.

**Phương pháp:** Dựa vào tỷ lệ chuyển đổi

**Công thức:**

## **6. Lọc dữ liệu vận tốc (Tùy chọn)**

****

## **7. Ghi chú và hiển thị vận tốc**

## 

# 2. DPI

## 2.1. DPI của Ảnh

DPI (Dots Per Inch) là đơn vị đo độ phân giải của ảnh, thường được sử dụng trong in ấn. Nó biểu thị số điểm ảnh trên mỗi inch của hình ảnh.

* **Ý nghĩa của DPI:**
  + **Cao (300 DPI trở lên):** Phù hợp cho in ấn chất lượng cao.
  + **Thấp (72-150 DPI):** Thường dùng cho hiển thị trên màn hình.
  + DPI không trực tiếp ảnh hưởng đến chất lượng ảnh kỹ thuật số, mà chỉ liên quan đến cách nó được hiển thị hoặc in.
* **Kiểm tra và thay đổi DPI:**Sử dụng các công cụ như Photoshop, GIMP hoặc phần mềm chỉnh sửa ảnh để xem và thay đổi thông số DPI.

## 2.2. Camera và Độ phân giải

Đối với camera, yếu tố quan trọng là:

#### **a. Độ phân giải cảm biến (Megapixel)**

* Đây là số lượng điểm ảnh (pixel) mà cảm biến camera có thể ghi nhận. Ví dụ:
  + **12MP:** Thường dùng trong điện thoại.
  + **20MP+**: Thường dùng trong máy ảnh chuyên nghiệp.

#### **b. Kích thước cảm biến**

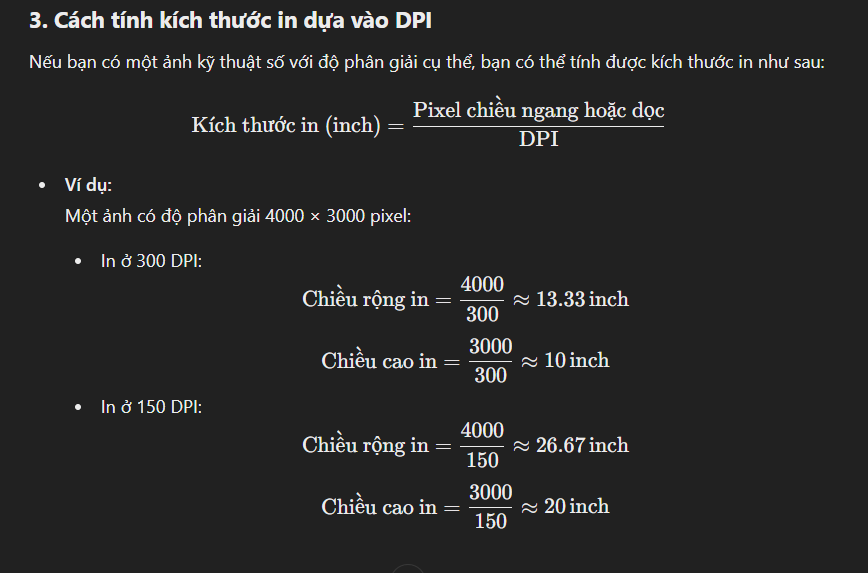
* Kích thước vật lý của cảm biến ảnh hưởng đến chất lượng, nhất là khi chụp thiếu sáng. Ví dụ:
  + **Full-frame (35mm):** Máy ảnh chuyên nghiệp.
  + **APS-C, Micro Four Thirds:** Máy ảnh nhỏ gọn.

#### **c. Ống kính và thông số kỹ thuật**

* **Tiêu cự (Focal Length):** Ảnh hưởng đến góc nhìn.
* **Khẩu độ (Aperture):** Quy định lượng ánh sáng đi vào cảm biến.

#### **d. Tốc độ và tính năng**

* **Tốc độ chụp (Shutter Speed):** Ảnh hưởng đến khả năng bắt chuyển động.
* **Chế độ ISO:** Ảnh hưởng đến độ nhạy sáng.



# 3. Chuyển đổi từ Inch sang Mét

Để chuyển đổi từ **inch** sang **mét**, bạn có thể sử dụng công thức sau:

Ví dụ:

* 10 inch × 0.0254 = 0.254 m
* 15 inch × 0.0254 = 0.381 m

# 4. Sử dụng CSharp để check DPI của ảnh và camera

## 4.1. Kiểm tra DPI của ảnh

Để kiểm tra DPI của ảnh trong C#, bạn có thể sử dụng thư viện System.Drawing. Đây là một thư viện cơ bản của .NET để làm việc với đồ họa, bao gồm cả đọc và xử lý ảnh.

**Cách kiểm tra DPI của ảnh:**

| using System;  using System.Drawing;  class Program  {  static void Main()  {  string imagePath = "path\_to\_your\_image.jpg"; // Đường dẫn tới ảnh của bạn  using (Bitmap bitmap = new Bitmap(imagePath))  {  float dpiX = bitmap.HorizontalResolution; // DPI theo chiều ngang  float dpiY = bitmap.VerticalResolution; // DPI theo chiều dọc    Console.WriteLine("DPI (Horizontal): " + dpiX);  Console.WriteLine("DPI (Vertical): " + dpiY);  }  }  } |
| --- |

* HorizontalResolution và VerticalResolution trả về giá trị DPI của ảnh theo chiều ngang và dọc. Thông thường, nếu ảnh không có thông tin DPI được lưu, các giá trị này có thể trả về mặc định (ví dụ: 96 DPI).

## 4.2. Kiểm tra DPI của Camera

Để kiểm tra DPI của camera, điều này không phải là một phép toán đơn giản như với ảnh. DPI của một camera không được xác định trực tiếp từ phần cứng như ảnh mà có thể phụ thuộc vào độ phân giải của ảnh mà camera chụp và cách hiển thị ảnh đó.

Trong trường hợp này, bạn có thể sử dụng các thư viện như **MediaCapture API** trong UWP (Universal Windows Platform) hoặc các thư viện bên ngoài như AForge.NET hoặc Emgu.CV để làm việc với camera.

**Cách kiểm tra thông tin về camera trong C#:**

Bạn có thể sử dụng thư viện Windows.Media.Capture trong UWP hoặc WinForms để lấy thông tin về camera.

| using System;  using Windows.Media.Capture;  using Windows.Storage;  using Windows.UI.Xaml.Controls;  using System.Threading.Tasks;  class Program  {  static async Task Main(string[] args)  {  // Lấy thông tin về camera (nếu dùng UWP)  var camera = await GetCameraInfoAsync();  Console.WriteLine("Camera Resolution: " + camera);  }  static async Task<string> GetCameraInfoAsync()  {  var mediaCapture = new MediaCapture();  await mediaCapture.InitializeAsync();    // Ví dụ lấy thông tin độ phân giải của camera  var resolution = mediaCapture.VideoDeviceController.GetAvailableMediaStreamProperties(Windows.Media.Capture.MediaStreamType.VideoPreview);  return resolution.ToString();  }  } |
| --- |

* Trong ví dụ trên, bạn cần phải triển khai một ứng dụng UWP hoặc WinForms để sử dụng các API của Windows để truy cập vào camera. Sau đó, bạn có thể kiểm tra độ phân giải và các thông số của camera.



# 5. Ứng dụng tính vận tốc của phương tiện

**Hướng đi của bài toán**

Bài toán này thuộc lĩnh vực **xử lý hình ảnh** và **thị giác máy tính**, với mục tiêu chính là **phát hiện, theo dõi và tính toán vận tốc của phương tiện** trong video. Hướng đi của bài toán được tổ chức theo các bước chính như sau:

## 1. Định nghĩa bài toán

* **Đầu vào**:
  + Một video giao thông.
  + Thông số chiều rộng làn đường thực tế (mét).
  + Thông tin về số frame cần bỏ qua (skip\_frames).
* **Đầu ra**:
  + Vận tốc của từng phương tiện được phát hiện trong video, theo đơn vị km/h.
  + Video đầu ra có chú thích ID và vận tốc của từng phương tiện.
* **Mục tiêu**:
  + Xác định vận tốc trung bình và vận tốc tối đa của từng phương tiện trong video.

## 2. Thành phần chính trong giải pháp

### 2.1. Phát hiện đối tượng (Object Detection)

* Sử dụng mô hình **YOLOv8** (từ thư viện ultralytics) để phát hiện các phương tiện.
* Mỗi phương tiện được biểu diễn bằng một **bounding box** (tọa độ x1, y1, x2, y2).
* Phân loại các đối tượng (xe hơi, xe tải, xe máy, v.v.) dựa trên nhãn (cls).

### 2.2. Theo dõi đối tượng (Object Tracking)

* Không sử dụng module YOLO tích hợp sẵn để theo dõi đối tượng, thay vào đó tự triển khai cơ chế gán ID cho từng phương tiện:
  + Mỗi phương tiện được so sánh với các đối tượng đã được phát hiện ở khung hình trước.
  + Gán ID mới nếu không có ID nào phù hợp (dựa trên khoảng cách giữa tâm bounding box).

### 2.3. Hiệu chuẩn camera (Calibration)

* Sử dụng chiều rộng thực tế của làn đường (mét) và chiều rộng của làn đường trong ảnh (pixel) để tính **tỷ lệ pixel/mét**:

### 2.4. Tính vận tốc

* Dựa trên vị trí của đối tượng trong các khung hình liên tiếp và thời gian thực giữa các khung hình:
  + Tính khoảng cách di chuyển (pixel) giữa hai khung hình.
  + Chuyển đổi từ pixel sang mét bằng meters\_per\_pixel.
  + Tính vận tốc theo công thức:

​

* + Chuyển đổi vận tốc từ m/s sang km/h:

### 2.5. Skip Frame (Tăng hiệu suất)

* Bỏ qua một số khung hình để giảm tải xử lý mà vẫn đảm bảo tính toán vận tốc chính xác:
  + Tăng thời gian thực giữa các khung hình bằng:

### 2.6. Xuất video đầu ra

* Ghi lại bounding box, ID, và vận tốc của từng phương tiện trong video đầu ra.
* Tạo trực quan giúp dễ dàng theo dõi kết quả.

## 3. Mã nguồn

**cal\_Velocity\_Skipframe.py**

| **import** cv2 **import** numpy **as** np **from** ultralytics **import** YOLO  **class** **SpeedDetector**:  **def** **\_\_init\_\_**(self, model\_path='yolov8n.pt', lane\_width\_meters=3.5, skip\_frames=2):  """  Khởi tạo detector  skip\_frames: số frame bỏ qua giữa mỗi lần xử lý  """  self.model = YOLO(model\_path)  self.lane\_width\_meters = lane\_width\_meters  self.vehicle\_tracking = {}  self.calibration\_done = **False**  self.meters\_per\_pixel = **None**  self.next\_id = 0  self.skip\_frames = skip\_frames    **def** **calculate\_speed**(self, pos1, pos2, time\_diff):  """  Tính vận tốc giữa 2 điểm với thời gian đã tính đến skipframe  """  **if** **not** self.calibration\_done:  **raise** ValueError("Cần hiệu chuẩn trước khi tính vận tốc!")    **if** time\_diff < 0.001:  **return** 0.0    dx = pos2[0] - pos1[0]  dy = pos2[1] - pos1[1]  distance\_pixels = np.sqrt(dx\*\*2 + dy\*\*2)  distance\_meters = distance\_pixels \* self.meters\_per\_pixel  speed\_mps = distance\_meters / time\_diff    *# Giới hạn tốc độ hợp lý (0-200 km/h)*  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):  **if** **not** self.calibration\_done:  self.meters\_per\_pixel = self.lane\_width\_meters / lane\_width\_pixels  self.calibration\_done = **True**    cap = cv2.VideoCapture(video\_path)  **if** **not** cap.isOpened():  **raise** ValueError("Không thể mở 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))    *# Thời gian thực giữa các frame được xử lý (tính đến skipframe)*  actual\_frame\_time = frame\_time \* (self.skip\_frames + 1)    fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  out = cv2.VideoWriter(output\_path, fourcc, fps/(self.skip\_frames + 1), (width, height))    frame\_count = 0  processed\_count = 0    **while** cap.isOpened():  ret, frame = cap.read()  **if** **not** ret:  **break**    frame\_count += 1    *# Skip frames*  **if** frame\_count % (self.skip\_frames + 1) != 0:  **continue**    processed\_count += 1    *# Detect đối tượng*  results = self.model(frame, stream=**True**)  current\_frame\_ids = set()    **for** result **in** results:  boxes = result.boxes  **for** box **in** boxes:  **if** int(box.cls[0]) **in** [2, 3, 5, 7]:  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)    *# Tìm ID phù hợp nhất dựa trên khoảng cách*  min\_dist = float('inf')  matched\_id = **None**    **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: *# Ngưỡng 100 pixel*  min\_dist = dist  matched\_id = obj\_id    **if** matched\_id **is** **None**:  matched\_id = self.next\_id  self.next\_id += 1  self.vehicle\_tracking[matched\_id] = {  'positions': [],  'timestamps': [],  'velocities': [],  'last\_update': processed\_count  }    current\_frame\_ids.add(matched\_id)  track = self.vehicle\_tracking[matched\_id]    *# Cập nhật tracking*  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)    *# Lọc nhiễu với trung bình động*  **if** len(track['velocities']) > 3:  speed = np.mean(track['velocities'][-3:])    *# Vẽ boundingbox và vận tốc*  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'ID: {matched\_id} Speed: {speed:.1f} km/h',   (int(x1), int(y1)-10),   cv2.FONT\_HERSHEY\_SIMPLEX,   0.5,   (0, 255, 0),   2)    *# Xóa tracking cũ*  **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]    out.write(frame)  cv2.imshow('Vehicle Speed Detection', frame)  **if** cv2.waitKey(1) & 0xFF == ord('q'):  **break**    cap.release()  out.release()  cv2.destroyAllWindows()    **return** self.vehicle\_tracking  **def** **main**():  *# Khởi tạo detector với skipframe*  detector = SpeedDetector(  lane\_width\_meters=3.5,  skip\_frames=3 *# Skip 2 frames, xử lý frame thứ 3*  )    video\_path = 'b.mp4'  tracking\_data = detector.process\_video(  video\_path=video\_path,  output\_path='output\_speed.mp4',  lane\_width\_pixels=243  )    print("\nThống kê vận tốc:")  **for** obj\_id, data **in** tracking\_data.items():  **if** len(data['velocities']) > 0:  velocities = [v **for** v **in** data['velocities'] **if** 0 <= v <= 200]  **if** velocities:  avg\_speed = np.mean(velocities)  max\_speed = np.max(velocities)  print(f"Xe {obj\_id}:")  print(f" Vận tốc trung bình: {avg\_speed:.1f} km/h")  print(f" Vận tốc cao nhất: {max\_speed:.1f} km/h")  **if** \_\_name\_\_ == "\_\_main\_\_":  main() |
| --- |