**BÁO CÁO CÔNG VIỆC**

**Công việc số:** 75

**Mô tả công việc:** Tìm hiểu thuật toán điều chỉnh độ nghiêng của biển số xe.

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 15/04/2025

**Ngày kết thúc:** 15/04/2025

***Ghi chú:***

**NỘI DUNG TÀI LIỆU**

[**1. Thuật toán Xoay Ảnh Biển Số Để Làm Thẳng (Deskew License Plate Image) 2**](#_9n35kgkxqj21)

[1.1. (Bước 1) Đọc và hiển thị ảnh gốc 3](#_zbh30p3r2kqc)

[1.2. (Bước 2) Chuyển ảnh sang ảnh xám (Grayscale) 3](#_749n9yl7y052)

[1.3. (Bước 3) Làm mờ ảnh (Gaussian Blur) 3](#_adcej9pkpi60)

[1.4. (Bước 4) Nhị phân hóa ảnh (Thresholding) 3](#_crp27qft0bev)

[1.5. (Bước 5) Tìm các contour 4](#_g6jtsdpae8vc)

[1.6. (Bước 6) Lấy contour lớn nhất (biển số) 4](#_jtbkbjh3fwym)

[1.7. (Bước 7) Tính hình chữ nhật bao quanh contour 4](#_1m7f3nfp9zb7)

[1.8. (Bước 8) Điều chỉnh góc nghiêng 5](#_fmsl6q6td8pv)

[1.9. (Bước 9) Xoay ảnh để làm thẳng 5](#_37m21rfmtdfc)

[1.9.1. Tính tâm ảnh: 5](#_u58us0jh64ts)

[1.9.2. Tạo ma trận xoay: 5](#_x1rm51alk3qw)

[1.9.3. Áp dụng xoay: 5](#_y97cqip61yhl)

[1.10. (Bước 10) Trả về ảnh đã làm thẳng 6](#_or0yanfh4yim)

[**2. Mã nguồn đầy đủ 6**](#_86zjgpwykycd)

[**3. Ghép vào file chính 7**](#_uolqwf3hithu)

# 

# 1. Thuật toán Xoay Ảnh Biển Số Để Làm Thẳng (Deskew License Plate Image)

**Mục tiêu:** Làm thẳng một ảnh biển số xe bị nghiêng (skewed) bằng cách phát hiện hướng nghiêng và xoay ảnh ngược lại góc lệch để trả về trạng thái thẳng.

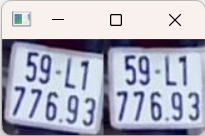
**Tổng quan các bước:**

1. Đọc ảnh và chuyển sang ảnh xám
2. Làm mờ ảnh để giảm nhiễu
3. Nhị phân hóa ảnh
4. Phát hiện các đường bao (contour)
5. Chọn contour lớn nhất (giả định là biển số)
6. Tính hình chữ nhật bao quanh contour
7. Tính góc nghiêng của biển số
8. Điều chỉnh góc để làm thẳng
9. Xoay ảnh dựa trên góc vừa tính
10. Trả về ảnh đã xoay

**Hình ảnh các bước của của thuật toán**



**Hình ảnh đối chiếu ban đầu và sau khi xoay:**



## 1.1. (Bước 1) Đọc và hiển thị ảnh gốc

* Sử dụng cv2.imread() để đọc ảnh biển số từ file.
* Hiển thị ảnh gốc để đối chiếu.

| image = cv2.imread("license\_plate.jpg") |
| --- |



## 1.2. (Bước 2) Chuyển ảnh sang ảnh xám (Grayscale)

* Ảnh màu RGB được chuyển sang ảnh xám để xử lý dễ hơn.

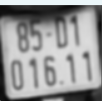
| gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) |
| --- |



## 1.3. (Bước 3) Làm mờ ảnh (Gaussian Blur)

* Làm mờ ảnh để loại bỏ nhiễu nhỏ không cần thiết trước khi phát hiện cạnh.

| blur = cv2.GaussianBlur(gray, (5, 5), 0) |
| --- |



## 1.4. (Bước 4) Nhị phân hóa ảnh (Thresholding)

* Sử dụng phương pháp cv2.THRESH\_BINARY + cv2.THRESH\_OTSU để tạo ảnh nhị phân trắng-đen.

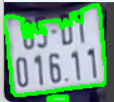
| \_, binary = cv2.threshold(blur, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU) |
| --- |



## 1.5. (Bước 5) Tìm các contour

* Sử dụng cv2.findContours() để tìm các vùng trắng nối liền (biển số nằm trong số này).

| contours, \_ = cv2.findContours(binary, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) |
| --- |



## 1.6. (Bước 6) Lấy contour lớn nhất (biển số)

* Giả định rằng contour có diện tích lớn nhất là biển số.

| largest\_contour = max(contours, key=cv2.contourArea) |
| --- |

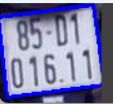


## 1.7. (Bước 7) Tính hình chữ nhật bao quanh contour

* Dùng cv2.minAreaRect() để tìm **hình chữ nhật xoay nhỏ nhất** bao quanh contour.

| rect = cv2.minAreaRect(largest\_contour) |
| --- |

* Trả về:  
   (center\_x, center\_y), (width, height), angle
* Trong đó:
  + (center\_x, center\_y): Tọa độ tâm của hình chữ nhật nhỏ nhất bao contour.
  + (width, height): Kích thước hình chữ nhật.
  + angle: Góc nghiêng của cạnh dài hoặc cạnh ngắn của hình chữ nhật so với trục Ox



## 1.8. (Bước 8) Điều chỉnh góc nghiêng

* Góc từ minAreaRect() là **góc giữa cạnh ngắn của hình chữ nhật với trục X**.

| if angle < -45:  angle = 90 + angle elif angle > 45:  angle = angle - 90 |
| --- |

* Nhằm đảm bảo góc nghiêng là so với **cạnh dài nằm ngang**, để sau đó có thể xoay ngược lại.

## 1.9. (Bước 9) Xoay ảnh để làm thẳng

### 1.9.1. Tính tâm ảnh:

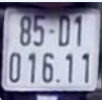
| (h, w) = image.shape[:2] center = (w // 2, h // 2) |
| --- |

### 1.9.2. Tạo ma trận xoay:

| M = cv2.getRotationMatrix2D(center, angle, 1.0) |
| --- |

### 1.9.3. Áp dụng xoay:

| rotated = cv2.warpAffine(  image,  M,  (w, h),  flags=cv2.INTER\_CUBIC,  borderMode=cv2.BORDER\_REPLICATE ) |
| --- |



## 1.10. (Bước 10) Trả về ảnh đã làm thẳng

* Ảnh đầu ra là ảnh đã được xoay đúng góc để biển số nằm ngang.
* Có thể hiển thị ảnh gốc và ảnh đã xoay để so sánh.

**🔎 Ưu điểm của thuật toán:**

* Tự động phát hiện hướng nghiêng của biển số.
* Không cần dữ liệu huấn luyện.
* Phù hợp với biển số đơn giản, rõ nét.

**⚠️ Hạn chế:**

* Không hoạt động tốt nếu contour không đại diện chính xác cho biển số.
* Cần hình ảnh có độ tương phản cao để tách contour hiệu quả.

**📌 Ứng dụng:**

* Làm sạch ảnh đầu vào trước khi nhận dạng ký tự biển số (OCR).
* Tăng độ chính xác của mô hình nhận diện biển số xe.

# 2. Mã nguồn đầy đủ

| import cv2 import numpy as np import os  def deskew\_license\_plate(image):  gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  blur = cv2.GaussianBlur(gray, (5, 5), 0)  \_, binary = cv2.threshold(blur, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  contours, \_ = cv2.findContours(binary, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)   if contours:  largest\_contour = max(contours, key=cv2.contourArea)  rect = cv2.minAreaRect(largest\_contour)  angle = rect[2]   if angle < -45:  angle += 90  elif angle > 45:  angle -= 90   (h, w) = image.shape[:2]  center = (w // 2, h // 2)  M = cv2.getRotationMatrix2D(center, angle, 1.0)  rotated = cv2.warpAffine(image, M, (w, h), flags=cv2.INTER\_CUBIC,  borderMode=cv2.BORDER\_REPLICATE)  return rotated  return image   input\_dir = "./Cropped" output\_dir = "./Cropped\_Deskewed"  # Tạo thư mục đầu ra nếu chưa tồn tại os.makedirs(output\_dir, exist\_ok=True)  # Lặp qua các file ảnh trong thư mục đầu vào for filename in os.listdir(input\_dir):  if filename.lower().endswith((".png", ".jpg", ".jpeg")):  image\_path = os.path.join(input\_dir, filename)  image = cv2.imread(image\_path)   if image is not None:  result = deskew\_license\_plate(image)  save\_path = os.path.join(output\_dir, filename)  cv2.imwrite(save\_path, result)  print(f"✔ Đã xử lý và lưu: {save\_path}")  else:  print(f"✘ Không thể đọc ảnh: {image\_path}") |
| --- |

# 3. Ghép vào file chính

| import os import cv2 import numpy as np import logging import warnings from skimage.filters import threshold\_local from skimage import measure import imutils from ultralytics import YOLO import tensorflow as tf from tensorflow.keras.models import load\_model import shutil from PIL import Image, ImageDraw, ImageFont   # ==== LOGGING & WARNING SETUP ==== # Tắt log TensorFlow từ C++ os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3' logging.getLogger('tensorflow').setLevel(logging.ERROR) logging.getLogger('absl').setLevel(logging.ERROR) logging.getLogger('ultralytics').setLevel(logging.ERROR) warnings.filterwarnings("ignore", category=UserWarning, module='tensorflow')   # ==== CONSTANTS ==== CROPPED\_DIR = r"./Test/Cropped" CHAR\_OUTPUT\_DIR = r"./Test/output\_chars" MODEL\_PATH = r"./Model/model\_nhan\_dang\_bien\_so\_final.h5" os.makedirs(CROPPED\_DIR, exist\_ok=True) os.makedirs(CHAR\_OUTPUT\_DIR, exist\_ok=True)  # ==== MODEL LOADING ==== yolo\_model = YOLO("./Model/License-plate-detection.pt")  # ==== UTILS ====  def convert\_to\_square(image, pad\_color=255):  """Pad image to make it square."""  h, w = image.shape[:2]  size\_diff = abs(h - w)  if h == w:  return image  if len(image.shape) == 2:  pad = [(size\_diff // 2, size\_diff - size\_diff // 2), (0, 0)] if h < w else [(0, 0), (size\_diff // 2, size\_diff - size\_diff // 2)]  else:  pad = [(0, 0), (0, 0), (0, 0)]  if h < w:  pad[0] = (size\_diff // 2, size\_diff - size\_diff // 2)  else:  pad[1] = (size\_diff // 2, size\_diff - size\_diff // 2)  return np.pad(image, pad, mode='constant', constant\_values=pad\_color)  def preprocess\_license\_plate(plate\_img):  """Convert plate image to binary using adaptive thresholding."""  hsv = cv2.cvtColor(plate\_img, cv2.COLOR\_BGR2HSV)  \_, \_, v = cv2.split(hsv)  T = threshold\_local(v, 15, offset=10, method="gaussian")  return ((v > T) \* 255).astype("uint8")  def extract\_and\_sort\_characters(thresh, plate\_img):  """Extract character candidates from binary image."""  inverted = cv2.bitwise\_not(thresh)  resized = imutils.resize(inverted, width=400)  blurred = cv2.medianBlur(resized, 5)  labels = measure.label(blurred, connectivity=2, background=0)   candidates = []  scale\_x = plate\_img.shape[1] / resized.shape[1]  scale\_y = plate\_img.shape[0] / resized.shape[0]   for label in np.unique(labels):  if label == 0:  continue  mask = (labels == label).astype("uint8") \* 255  contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  if not contours:  continue   contour = max(contours, key=cv2.contourArea)  x, y, w, h = cv2.boundingRect(contour)  aspect\_ratio = w / float(h)  solidity = cv2.contourArea(contour) / float(w \* h)  height\_ratio = h / float(plate\_img.shape[0])   if 0.1 < aspect\_ratio < 1.0 and solidity > 0.1 and 0.35 < height\_ratio < 2.0:  x\_orig, y\_orig = int(x \* scale\_x), int(y \* scale\_y)  w\_orig, h\_orig = int(w \* scale\_x), int(h \* scale\_y)  char\_img = plate\_img[y\_orig:y\_orig + h\_orig, x\_orig:x\_orig + w\_orig]  square\_char = convert\_to\_square(char\_img)  resized\_char = cv2.resize(square\_char, (28, 28), cv2.INTER\_AREA)  candidates.append((resized\_char, (y, x)))   # Sort by rows then columns  candidates = sorted(candidates, key=lambda x: x[1][0])  mid\_y = np.median([pos[0] for \_, pos in candidates])  top = sorted([c for c in candidates if c[1][0] < mid\_y], key=lambda x: x[1][1])  bottom = sorted([c for c in candidates if c[1][0] >= mid\_y], key=lambda x: x[1][1])  return top + bottom  def save\_characters(candidates, original\_filename):  """Save sorted character images to output directory."""  prefix = os.path.splitext(os.path.basename(original\_filename))[0]  for idx, (char\_img, \_) in enumerate(candidates):  filename = os.path.join(CHAR\_OUTPUT\_DIR, f"{prefix}\_char\_{idx:02d}.png")  cv2.imwrite(filename, char\_img)  def detect\_and\_crop\_license\_plate(image\_path):  """Detect and crop license plates from an image."""  img = cv2.imread(image\_path)  if img is None:  print(f"⚠️ Cannot read image: {image\_path}")  return []   results = yolo\_model(image\_path)  cropped\_paths = []   for i, result in enumerate(results):  for j, box in enumerate(result.boxes):  x1, y1, x2, y2 = map(int, box.xyxy[0])  plate = img[y1:y2, x1:x2]  if plate.size == 0:  continue  filename = os.path.splitext(os.path.basename(image\_path))[0]  cropped\_path = os.path.join(CROPPED\_DIR, f"{filename}\_plate\_{i}\_{j}.jpg")  cv2.imwrite(cropped\_path, plate)  # print(f"✅ Cropped plate saved: {cropped\_path}")  cropped\_paths.append(cropped\_path)   return cropped\_paths   **def deskew\_license\_plate(image):  gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  blur = cv2.GaussianBlur(gray, (5, 5), 0)  \_, binary = cv2.threshold(blur, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  contours, \_ = cv2.findContours(binary, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)   if contours:  largest\_contour = max(contours, key=cv2.contourArea)  rect = cv2.minAreaRect(largest\_contour)  angle = rect[2]   if angle < -45:  angle += 90  elif angle > 45:  angle -= 90   (h, w) = image.shape[:2]  center = (w // 2, h // 2)  M = cv2.getRotationMatrix2D(center, angle, 1.0)  rotated = cv2.warpAffine(image, M, (w, h), flags=cv2.INTER\_CUBIC,  borderMode=cv2.BORDER\_REPLICATE)  return rotated  return image**  def process\_single\_image(image\_path):  """Full pipeline for a single image: detect plate, crop, segment characters, recognize and draw result."""  img = cv2.imread(image\_path)  if img is None:  print(f"⚠️ Cannot read image: {image\_path}")  return   results = yolo\_model(image\_path)  for i, result in enumerate(results):  for j, box in enumerate(result.boxes):  x1, y1, x2, y2 = map(int, box.xyxy[0])  plate = img[y1:y2, x1:x2]  if plate.size == 0:  continue    # Deskew the license plate image   **plate = deskew\_license\_plate(plate)**     # Tiền xử lý & tách ký tự  binary = preprocess\_license\_plate(plate)  candidates = extract\_and\_sort\_characters(binary, plate)  prefix = f"plate\_{i}\_{j}"  for idx, (char\_img, \_) in enumerate(candidates):  filename = os.path.join(CHAR\_OUTPUT\_DIR, f"{prefix}\_char\_{idx:02d}.png")  cv2.imwrite(filename, char\_img)   # Nhận diện ký tự  recognized\_text = ""  image\_paths = sorted([  f for f in os.listdir(CHAR\_OUTPUT\_DIR)  if f.startswith(prefix) and f.endswith(".png")  ])  for img\_file in image\_paths:  full\_path = os.path.join(CHAR\_OUTPUT\_DIR, img\_file)  ky\_tu, \_ = nhan\_dien\_ky\_tu(full\_path, MODEL\_PATH)  recognized\_text += ky\_tu   print(f"✅ Biển số phát hiện: {recognized\_text}")   # Vẽ biển số lên ảnh  img\_pil = Image.fromarray(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))  draw = ImageDraw.Draw(img\_pil)  try:  font = ImageFont.truetype("arial.ttf", 32)  except:  font = ImageFont.load\_default()  draw.text((x1, y1 - 40), recognized\_text, font=font, fill=(255, 0, 0))   # Vẽ khung biển số  draw.rectangle([(x1, y1), (x2, y2)], outline=(0, 255, 0), width=3)  img\_result = cv2.cvtColor(np.array(img\_pil), cv2.COLOR\_RGB2BGR)   # Hiển thị ảnh  cv2.imshow("Ket qua nhan dien bien so bang Deep Learning", img\_result)  cv2.waitKey(0)  cv2.destroyAllWindows()   def nhan\_dien\_ky\_tu(duong\_dan\_anh, model\_path, img\_width=64, img\_height=64, grayscale=True):  """Nhận diện ký tự từ ảnh bằng model đã huấn luyện."""  model = load\_model(model\_path)   if grayscale:  img = cv2.imread(duong\_dan\_anh, cv2.IMREAD\_GRAYSCALE)  img = cv2.resize(img, (img\_width, img\_height))  img = np.expand\_dims(img, axis=-1)  else:  img = cv2.imread(duong\_dan\_anh)  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  img = cv2.resize(img, (img\_width, img\_height))   img = img.astype('float32') / 255.0  img = np.expand\_dims(img, axis=0)   prediction = model.predict(img, verbose=0)  predicted\_class = np.argmax(prediction, axis=1)[0]  probability = np.max(prediction)   idx\_to\_label = {  0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9',  10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F', 16: 'G', 17: 'H',  18: 'K', 19: 'L', 20: 'M', 21: 'N', 22: 'P', 23: 'R', 24: 'S', 25: 'T',  26: 'U', 27: 'V', 28: 'X', 29: 'Y', 30: 'Z'  }   return idx\_to\_label[predicted\_class], probability  def recognize():  """Nhận diện toàn bộ ký tự từ thư mục ./output\_chars."""  image\_paths = sorted([  f for f in os.listdir(CHAR\_OUTPUT\_DIR)  if f.lower().endswith(('.png', '.jpg', '.jpeg'))  ])  license\_plate = ""  for img in image\_paths:  full\_path = os.path.join(CHAR\_OUTPUT\_DIR, img)  ky\_tu, \_ = nhan\_dien\_ky\_tu(full\_path, MODEL\_PATH)  license\_plate += ky\_tu  print(f"Biển số nhận diện được: {license\_plate}")  # ==== MAIN ==== if \_\_name\_\_ == "\_\_main\_\_":  test\_image = r"./0000\_02187\_b.jpg"  process\_single\_image(test\_image)   # Cleanup  shutil.rmtree(CHAR\_OUTPUT\_DIR)  shutil.rmtree(CROPPED\_DIR) |
| --- |

Kết quả trước hiệu chỉnh



Kết quả sau hiệu chỉnh:



1. **Trước khi hiệu chỉnh (Ảnh 1)**:
   * Biển số xe bị biến dạng một phần do góc chụp của camera. Các ký tự trên biển số có vẻ bị kéo dài và không thẳng hàng.
   * Việc này khiến cho văn bản khó đọc, dẫn đến khả năng nhận diện sai hoặc sai sót trong việc trích xuất dữ liệu.
2. **Sau khi hiệu chỉnh (Ảnh 2)**:
   * Biển số xe đã được chỉnh sửa, xoay hoặc điều chỉnh sao cho thẳng đứng hơn, giúp các ký tự trở nên rõ ràng hơn. (Tuy nhiên hiển thị lên để minh họa vẫn sẽ dung ảnh gốc)
   * Văn bản trên biển số dễ đọc và thẳng hàng hơn, cải thiện độ chính xác khi nhận diện. Kết quả này cho thấy sự cải thiện rõ rệt trong khả năng trích xuất và xử lý dữ liệu.

**Kết luận**: Việc hiệu chỉnh góc nghiêng đã cải thiện đáng kể độ rõ ràng của biển số, giúp nhận diện chính xác hơn. Tuy nhiên ký tự nhận diện đã chuẩn theo thứ tự hàng trên trước nhưng kết quả nhận diện ký tự vẫn chưa chính xác lắm. Ta cần bổ sung lại dữ liệu huấn luyện hoặc hiệu chỉnh lại việc cắt ký tự sao cho không bị bắt vào 1 phần của ký tự bên cạnh