ชื่อหัวข้อ License Plate Recognition โปรแกรมอ่านป้ายทะเบียนรถยนต์

สมาชิก

นายคุณากร กัลยาวุฒิพงศ์ รหัสนิสิต 6621600143
นางสาวอัสวาณี อารง รหัสนิสิต 6621604700

วัตถุประสงค์

1. เรียนรู้การพัฒนาโปรแกรมเพื่อการตรวจจับป้ายทะเบียนรถยนต์
2. เป็นการฝึกออกแบบกระบวนการแปลงภาพเป็นข้อความตัวอักษร
3. เป็นการฝึกการเขียนโปรแกรมด้วยภาษา Python

อธิบายขั้นตอนวิธีการดำเนินงานโดยละเอียด

1. แปลงภาพเป็น binary ด้วยฟังก์ชั่น binarize_image
   - แปลงภาพเป็นสีเทา ใช้ grayscale_image = image.convert('L') (L หมายถึงโหมด Grayscale)
   - เบลอภาพด้วย Gaussian Blur ทำให้ภาพเนียนขึ้น ลด noise ที่ภาพ
     Gaussian Blur blurred_image = cv2.GaussianBlur(gray_array, (5, 5), 0)
   - แปลงเป็นภาพไบนารี binary_image = Image.fromarray(binary_array.astype(np.uint8) * 255)
     โดยการคูณ 255 จะมีแค่ภาพสีขาวดำเท่านั้นในภาพ

2. ทำ vertical_projection ด้วยฟังก์ชั่น vertical_projection
   - แปลงภาพไบนารีเป็น Array ทำให้ไปคำนวณได้ง่ายขึ้นและเร็วด้วย numpy
     img_array = np.array(binary_image)
   - คำนวณผลรวมของพิกเซลสีดำในแต่ละคอลัมน์ใช้เงื่อนไข img_array == 0 เอาสีดำ และรวมค่าของแต่ละคอลัมน์ใช้ np.sum(..., axis=0) จะได้ array ที่เก็บ pixel สีดำในแต่ละคอลัมน์ของภาพ
   - ผลลัพธ์ของ Vertical Projection เป็น Array แสดงจำนวน pixel สีดำในแต่ละ คอลัมน์ คอลัมน์ที่มีค่าพิกเซลสีดำเป็น 0 อาจจะเป็นช่องว่างระหว่างตัวอักษร ข้อมูลนี้สามารถนำไปใช้ในการบอกขอบเขตตัวอักษรในภาพได้

3. แยกตัวอักษรแต่ละตัวจากการทำ vertical_projection ด้วยฟังก์ชั่น segment_characters_modified
   - function segment_characters_modified สร้าง list เพื่อเก็บตัวอักษร
   - เดินดูทุกคอลัมน์ เริ่มต้นให้เป็น false ถ้าเจอ pixel สีดำ ถือว่าเจอขอบตัวอักษรแล้ว ให้เก็บไว้เป็น start_index ให้เป็น true เพื่อบอกว่าอยู่ในช่วงตัวอักษรแล้ว ถ้าค่า pixel ต่ำกว่าค่า T ให้เป็น end_index ไปเป็น false แล้วนำ start กับ end ไปเก็บไว้ใน list

4. ตัดตัวอักษรออกมาทีละตัว ด้วยช้ฟังก์ชั่น extract_characters โดยฟังก์ชั่นมีการทำงานดังนี้
   binary_image คือ ภาพขาว-ดำ segments คือ ตำแหน่งเริ่มต้นและสิ้นสุดของแต่ละตัวอักษร
   - ตัดแยกตัวอักษรแต่ละตัวโดย การวนลูปผ่านแต่ละ segment
   - ตัดเฉพาะส่วนของภาพตามความกว้างที่กำหนด ครอบตัดให้เหลือแค่ pixel ที่มีสีดำอยู่ return เป็นตัวอักษรแต่ละตัว

5. สร้าง pattern ตัวอักษร ด้วยฟังก์ชั่น get_character_pattern และ load_character_patterns
   get_character_pattern คือ การสร้าง pattern ของตัวอักษร โดยฟังก์ชั่นมีการทำงานดังนี้
   - แปลงภาพเป็น grayscale ปรับขนาดให้เท่ากัน และ แปลงเป็นภาพขาว-ดำ
   - คำนวณ vertical projection รวมค่าพิกเซลตามแนวตั้งของภาพ เพราะทำการครปตัดเฉพาะส่วนที่มีสีดำแล้ว พร้อม normalize ค่า projection

   load_character_patterns คือ สร้างฐานข้อมูลรูปแบบของตัวอักษรแต่ละตัว
   - กำหนด dictionary ของตัวอักษรและ path ของไฟล์รูปภาพ สร้าง pattern สำหรับทุกตัวอักษร วนลูปผ่านทุกตัวอักษร ใช้ get_character_pattern เพื่อสร้าง pattern

6. เทียบตัวอักษร ด้วยฟังก์ชั่น recognize_characters
   - วนลูปผ่านแต่ละ segment ตัดเอาเฉพาะส่วน projection ของตัวอักษรนั้นๆ normalize ค่าให้อยู่ในช่วง 0-1
   - เปรียบเทียบ projection กับ pattern ของทุกตัวอักษร คำนวณค่าความแตกต่าง (ยิ่งน้อยยิ่งเหมือน) np.abs(normalized_projection - pattern)
   - ตรวจสอบความเหมือน ถ้าค่าความแตกต่างน้อยกว่า threshold (0.5) ถือว่าเป็นตัวอักษรที่ถูกต้อง ส่งคืนรายการของตัวอักษรที่เหมือนที่สุด