รายงาน

Mini Project

ระบบตรวจจับเลน

จัดทำโดย

นาย พันธกานต์ คงทอง 6414631036

เสนอ

อาจารย์ นายคัมภีร์ ธีระเวช

รายงานนี้เป็นส่วนหนึ่งของวิชา การประมวลผลภาพดิจิทัล

ภาคเรียนที่ 2 ปี การศึกษา 2566

มหาวิทยาลัยราชภัฏร าไพพรรณี จันทบุรี

คณะวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

สารบัญ

เกี่ยวกับการตรวจจับเลนถนน

โปรแกรมนี้เป็นโปรแกรมที่สร้างขึ้นเพื่อตรวจจับการออกนอกเลนถนน โดยโปรแกรมนี้ทำงานด้วยการใช้วิดีโอของถนนและการตรวจจับสีในช่วง HSV ซึ่งช่วยในการตรวจจับเส้นทางของถนนในสภาวะแสงและเงาต่างๆ และยังมีการแสดงผลลัพธ์ในแต่ละขั้นตอนของการประมวลผลในหน้าต่างของโปรแกรมเพื่อให้ผสามารถติดตามการทำงานได้อย่างชัดเจนและทราบสถานะของรถว่าอยู่ในเส้นทางถูกต้องหรือไม่ และแจ้งเตือนเมื่อรถออกนอกเส้นทางของถนน

```python
import cv2
import numpy as np

vidcap = cv2.VideoCapture("Lane.mp4")
success, image = vidcap.read()

def nothing(x):
    pass

cv2.namedWindow("Trackbars")
cv2.createTrackbar("L - H", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("L - S", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("L - V", "Trackbars", 200, 255, nothing)
cv2.createTrackbar("U - H", "Trackbars", 255, 255, nothing)
cv2.createTrackbar("U - S", "Trackbars", 50, 255, nothing)
cv2.createTrackbar("U - V", "Trackbars", 255, 255, nothing)
```

ในโค้ดส่วนด้านบน จะมีการเรียกใช้งานฟังค์ชันต่างๆ

import cv 2 : เรียกใช้งาน OpenCV library.

import numpy as np: เรียกใช้งาน NumPy library ใช้สำหรับการประมวลผลทางคณิตศาสตร์และการทำงานกับข้อมูลตารางหรืออาร์เรย์

อ่านวิดีโอและเฟรม: โดยใช้ cv2.VideoCapture("Lane.mp4") เพื่อสร้างอ็อบเจกต์ vidcap เพื่ออ่านวิดีโอจากไฟล์ "Lane.mp4" และ vidcap.read() เพื่ออ่านเฟรมแรกของวิดีโอและบันทึกเฟรมในตัวแปร image รวมถึงค่าสำเร็จหรือไม่สำเร็จในการอ่านใน success.

สร้างและกำหนดค่าแทร็กบาร์: cv2.namedWindow("Trackbars") ใช้เพื่อสร้างหน้าต่างที่มีชื่อว่า "Trackbars" เพื่อใช้สำหรับการปรับค่าสีในรูปแบบ HSV (Hue, Saturation, Value) และ cv2.createTrackbar() ใช้สำหรับสร้างแทร็กบาร์และกำหนดค่าสีที่ต้องการตรวจจับ.

"L - H": ค่าล่างของสี Hue (ความสว่างของสีแดง).

"L - S": ค่าล่างของความเข้มของสี Saturation (ความอิ่มตัวของสี).

"L - V": ค่าล่างของความสว่างของสี Value (ความสว่างของภาพ).

"U - H": ค่าบนของสี Hue (ความสว่างของสีแดง).

"U - S": ค่าบนของความเข้มของสี Saturation (ความอิ่มตัวของสี).

"U - V": ค่าบนของความสว่างของสี Value (ความสว่างของภาพ).

```
left_base = 220
right_base = 460
while success:
    success, image = vidcap.read()
    frame = cv2.resize(image, (640, 480))


    tl = (222,387)
    bl = (70 ,472)
    tr = (400,380)
    br = (538,472)


    pts1 = np.float32([tl, bl, tr, br])
    pts2 = np.float32([[0, 0], [0, 480], [640, 0], [640, 480]])
```

การปรับมุมมองภาพ

ในโค้ดด้านบนมีการใช้งานอัลกอริทึม perspective transformation เพื่อทำให้ภาพอยู่ในมุมมองที่สามารถเห็นเส้นถนนได้อย่างชัดเจน

กำหนดตำแหน่งเริ่มต้นของจุดฐานซ้ายและขวา (left_base = 220 และ right_base = 460): กำหนดตำแหน่งเริ่มต้นของจุดฐานสำหรับการค้นหาเส้นทางซ้ายและขวาในแต่ละชั้นของภาพ.

จะกำหนดจุดสี่เหลี่ยม 4 จุด (tl, bl, tr, br) บนภาพเพื่อบอกถึงกรอบสี่เหลี่ยมที่เราต้องการที่จะแปลงให้อยู่ในมุมมองที่ถูกต้อง เพื่อให้เห็นถนนในมุมมองที่ตั้งแล้ว เราจะใช้จุด tl และ tr เป็นขอบขวางบนของถนน และ bl และ br เป็นขอบล่างของถนน.

หลังจากกำหนดจุดต่าง ๆ เรียบร้อยแล้ว ใช้ cv2.getPerspectiveTransform() เพื่อคำนวณ matrix ที่ใช้ในการแปลงภาพ โดยรับจุดเริ่มต้น (pts1) และจุดปลายทาง (pts2) ที่กำหนดไว้เพื่อแปลงภาพ.

การแปลงภาพ: ทำการใช้ cv2.warpPerspective() เพื่อทำการแปลงภาพตาม matrix ที่ได้จาก cv2.getPerspectiveTransform() โดยระบุขนาดของภาพที่ต้องการใหม่ (640x480) เพื่อให้เห็นถนนในมุมมองที่ถูกต้อง.

วนลูปการอ่านเฟรมและการปรับขนาดภาพ: โดยใช้ while success: เพื่อวนลูปจนกว่าการอ่านเฟรมจะไม่สำเร็จ และใช้ vidcap.read() เพื่ออ่านเฟรมถัดไปจากวิดีโอ จากนั้นใช้ cv2.resize() เพื่อปรับขนาดภาพให้อยู่ในขนาดที่กำหนด (640x480) เพื่อให้ง่ายต่อการประมวลผลต่อไป.

```
matrix = cv2.getPerspectiveTransform(pts1, pts2)
transformed_frame = cv2.warpPerspective(frame, matrix, (640,480))

hsv_transformed_frame = cv2.cvtColor(transformed_frame, cv2.COLOR_BGR2HSV)

l_h = cv2.getTrackbarPos("L - H", "Trackbars")
l_s = cv2.getTrackbarPos("L - S", "Trackbars")
l_v = cv2.getTrackbarPos("L - V", "Trackbars")
u_h = cv2.getTrackbarPos("U - H", "Trackbars")
u_s = cv2.getTrackbarPos("U - S", "Trackbars")
u_v = cv2.getTrackbarPos("U - V", "Trackbars")

lower = np.array([l_h,l_s,l_v])
upper = np.array([u_h,u_s,u_v])
lane_mask = cv2.inRange(hsv_transformed_frame, lower, upper)
mask = cv2.inRange(hsv_transformed_frame, lower, upper)
```

การแปลงเป็นมุมมองด้วย Perspective Transform

matrix = cv2.getPerspectiveTransform(pts1, pts2): ใช้ฟังก์ชัน cv2.getPerspectiveTransform() เพื่อคำนวณ matrix ที่ใช้ในการแปลงเพอร์สเปกที่ต้องการ โดยรับจุดเริ่มต้น (pts1) และจุดปลายทาง (pts2) ที่กำหนดไว้.

transformed_frame = cv2.warpPerspective(frame, matrix, (640,480)): ใช้ cv2.warpPerspective() เพื่อทำการแปลงภาพตาม matrix ที่ได้จาก cv2.getPerspectiveTransform() เพื่อให้ได้ภาพที่มีมุมมองที่ถูกต้องสำหรับการตรวจจับเส้นทาง โดยระบุขนาดของภาพที่ต้องการใหม่เป็น (640, 480).

การแปลงเป็นภาพ HSV:

hsv_transformed_frame = cv2.cvtColor(transformed_frame, cv2.COLOR_BGR2HSV): ใช้ cv2.cvtColor() เพื่อแปลงภาพที่ได้จากการแปลงเป็นมุมมองที่ถูกต้องเป็นรูปแบบสี HSV (Hue, Saturation, Value) เพื่อให้ง่ายต่อการตรวจจับสี.

การปรับค่าสีด้วยแทร็กบาร์:

โค้ดใช้แทร็กบาร์ที่สร้างขึ้นก่อนหน้านี้ (โดยใช้ cv2.createTrackbar()) เพื่อปรับค่าสีในรูปแบบ HSV ที่แปลงแล้ว เพื่อให้ผู้ใช้สามารถปรับแต่งพารามิเตอร์สีต่าง ๆ ได้ตามต้องการ ค่าสีที่ปรับได้ประกอบด้วย L - H, L - S, L - V (ค่าสีต่ำสุดสำหรับสีสีหลัก), U - H, U - S, U - V (ค่าสีสูงสุดสำหรับสีสีหลัก) โดยที่ L หมายถึง lower และ U หมายถึง upper สำหรับช่วงสีของตำแหน่งที่กำหนด.

การสร้าง Mask สำหรับการตรวจจับ:

โดยใช้ cv2.inRange() เพื่อสร้าง mask ที่มีเฉพาะสีที่ต้องการตรวจจับ โดยใช้ค่าสีที่ได้จากการปรับค่าสีของภาพ HSV ที่แปลงแล้ว. ผลลัพธ์คือ mask ที่เป็นขาวที่มีสีตามช่วงที่กำหนดและสีดำที่อื่น ๆ.

```python
# คางหมู
x_top_left = 220
y_top_left = 250
x_bottom_left = 190
y_bottom_left = 472
x_top_right = 460
y_top_right = 250
x_bottom_right = 490
y_bottom_right = 472

pts = np.array([[x_top_left, y_top_left],
                [x_bottom_left, y_bottom_left],
                [x_bottom_right, y_bottom_right],
                [x_top_right, y_top_right]], np.int32)
pts = pts.reshape((-1, 1, 2))

cv2.polylines(transformed_frame, [pts], isClosed=True, color=(255, 255, 255), thickness=2)

# เช็ค
if np.any(lane_mask[y_bottom_left:y_top_left, x_bottom_left:x_bottom_right] == 255) or \
    np.any(lane_mask[y_top_left:y_bottom_left, x_top_left:x_top_right] == 255):
# แจ้งเตือน
    cv2.putText(transformed_frame, 'Out of Lane!', (x_bottom_left, y_bottom_left - 250), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)
```

กำหนดพิกัดของจุดสี่เหลี่ยม

กำหนดพิกัดของจุดสี่เหลี่ยมที่ใช้เป็นตัวบ่งชี้ในการกำหนดพื้นที่ที่สนใจบนภาพ โดยทั้งหมดจะประกอบไปด้วย 4 จุด: (x_top_left, y_top_left), (x_bottom_left, y_bottom_left), (x_bottom_right, y_bottom_right), และ (x_top_right, y_top_right) โดยจุดนี้จะสร้างรูปสี่เหลี่ยมที่เรียกว่า trapezoid หรือคางหมูบนภาพ.

สร้าง trapezoid:

ใช้ np.array() เพื่อสร้าง array ของจุดในรูปแบบของ numpy array โดยให้สี่เหลี่ยมมีลักษณะเดิมที่กำหนดไว้ และใช้ cv2.polylines() เพื่อวาดเส้นต่อเขตของ trapezoid บนภาพที่ได้ทำการแปลงมุมมองแล้ว โดยตัวอาร์กิวเมนต์ที่สำคัญคือ transformed_frame เป็นภาพที่ได้จากการแปลงมุมมองแล้ว.

เช็คพื้นที่สนใจใน lane mask:

ใช้การเข้าถึงข้อมูลของ lane mask โดยตัดเอาเฉพาะพื้นที่สนใจที่กำหนดไว้ใน trapezoid เพื่อตรวจสอบว่ามีส่วนใดๆ ของ lane mask ที่มีค่าสีเป็น 255 (สีขาว) หรือไม่ ซึ่งหมายถึงการตรวจจับว่ามีส่วนของถนนอยู่ในพื้นที่นั้นหรือไม่.

แสดงข้อความแจ้งเตือน:

หากพบว่ามีส่วนของถนนอยู่ในพื้นที่ที่สนใจตาม trapezoid ที่กำหนด โค้ดจะแสดงข้อความ "Out of Lane!" บนภาพที่แปลงมุมมองแล้ว เพื่อแสดงข้อความเตือนว่ามีการออกนอกเส้นทางที่กำหนดไว้ในโปรแกรมนี้.

```
cv2.imshow("Black", lane_mask)
cv2.imshow("Lane Detection - Image Thresholding", transformed_frame)

if cv2.waitKey(10) == 27:
    break
```

การแสดงภาพที่ผ่านการประมวลผล

cv2.imshow() เป็นฟังก์ชันในไลบรารี OpenCV ที่ใช้สำหรับแสดงภาพหรือวิดีโอในหน้าต่าง GUI (Graphic User Interface) บนหน้าจอคอมพิวเตอร์

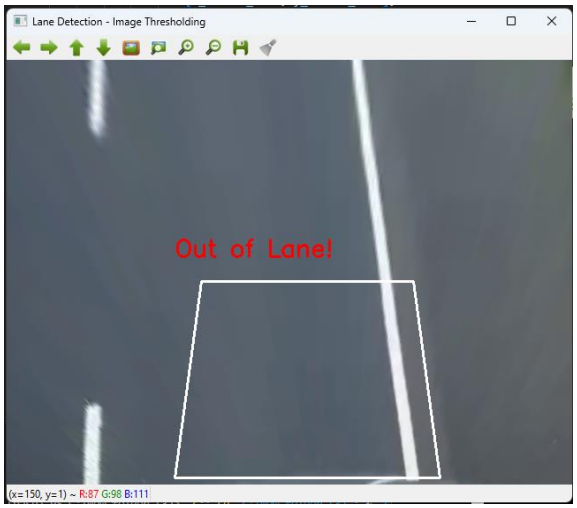
"Black": แสดงภาพที่ผ่านการแปลงสีและการปรับมุมมองด้วย Perspective Transform ในภาพนี้จะแสดงเฉพาะส่วนที่ถูกจับไว้โดย Thresholding ซึ่งจะเป็นสีขาว (255) ในส่วนที่ตรงกับสีที่กำหนดใน Trackbars และสีดำ (0) ในส่วนอื่นๆ

"Lane Detection - Image Thresholding": แสดงภาพที่ผ่านการปรับมุมมองและการแปลงสีด้วย Perspective Transform โดยภาพนี้จะแสดงเฉพาะเส้นทางของถนนที่ถูกจับและตรวจจับไว้ และมีการแสดงข้อความ "Out of Lane!" ถ้าพบว่ายานยนต์อยู่นอกเส้นทางของถนน

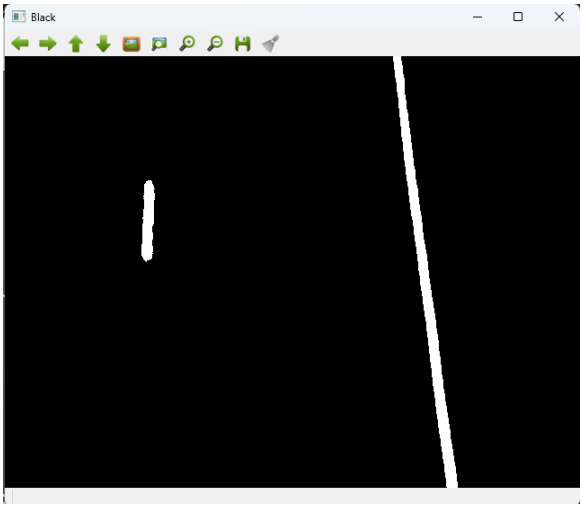
การรอรับคีย์บอร์ดด้วย cv2.waitKey(10) มีไว้เพื่อรอรับคีย์บอร์ดจากผู้ใช้ และถ้ามีการกดปุ่ม Esc (key code 27) โปรแกรมจะสิ้นสุดการทำงานและปิดหน้าต่างที่เปิดอยู่ทันที

ภาพที่แสดง

Lane Detection



Black



Trackbars