Assignment Robomaster สนามที่ 2

สมาชิกกลุ่ม LOCALHOST

1. 6610110425 คีตศิลป์ คงสี
2. 6610110034 คุณานนต์ หนูแสง
3. 6610110327 สิรวิชญ์ น้อยผา
4. 6610110341 สุธินันท์ รองพล

แปลงภาพและสร้าง Mask

ผู้รับผิดชอบ: 6610110034 คุณานนต์ หนูแสง

ส่วนนี้เป็นส่วนที่เกี่ยวกับการดึงภาพจากกล้องและแปลงสีเพื่อสร้าง Mask สำหรับการตรวจจับวัตถุ โดยแบ่งการทำงานได้ดังนี้

การอ่านภาพจากกล้อง

```
frame = ep_camera.read_cv2_image(strategy="newest", timeout=0.5)
```

- ep_camera.read_cv2_image() : เป็นฟังก์ชันที่ใช้ในการดึงภาพจากกล้องของหุ่นยนต์ ซึ่งใช้ OpenCV ในการประมวลผลภาพ
- strategy="newest" : หมายถึงการดึงภาพที่ใหม่ที่สุดจากกล้อง
- timeout=0.5 : หมายถึงการรอรับข้อมูลจากกล้องโดยให้เวลาสูงสุด 0.5 วินาที

การแปลงภาพจาก BGR เป็น HSV เพื่อใช้ในการวิเคราะห์สีของวัตถุได้แก่ลูกไก่ และขวดน้ำ

```
hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

- ใช้ฟังก์ชัน cv2.cvtColor เพื่อแปลงภาพจาก RGB เป็น HSV (Hue, Saturation, Value)
- การใช้พื้นที่สี HSV ช่วยให้การตรวจจับสีในภาพทำได้ง่ายและแม่นยำกว่าแบบ RGB เนื่องจากแบบ HSV สามารถแยกแยะความแตกต่างของสีโดยอิงตามเฉดสี (Hue) ได้ดีกว่าการรวมสามแม่สีอย่าง RGB

การกำหนดช่วงสีที่ต้องการตรวจจับ (Chicken และ Bottle)

สำหรับลูกไก่

```
lower_chicken = np.array([33, 150, 100])

upper_chicken = np.array([38, 255, 255])
```

lower_chicken และ upper_chicken เป็นค่าช่วงของสี HSV ที่กำหนดเพื่อใช้ในการตรวจจับวัตถุที่เป็นลูกไก่

- lower_chicken ระบุค่าสี HSV ต่ำสุดที่ต้องการตรวจจับ (Hue: 33, Saturation: 150, Value: 100)
- upper_chicken ระบุค่าสี HSV สูงสุด (Hue: 38, Saturation: 255, Value: 255)

ช่วงสีเหล่านี้ทำให้หุ่นยนต์สามารถมองเห็นลูกไก่ที่มีสีอยู่ในช่วงนี้ได้

สำหรับขวดน้ำ (ฉลากขวดน้ำ)

```
lower_bottle = np.array([95, 80, 100])

upper_bottle = np.array([120, 255, 255])
```

lower_bottle และ upper_bottle เป็นค่าช่วงของสี HSV ที่กำหนดเพื่อใช้ในการตรวจจับวัตถุที่เป็นฉลากขวดน้ำ เนื่องจากขวดน้ำเป็นพลาสติกใสจึงต้องใช้ฉลากขวดน้ำช่วยในการตรวจจับ

- lower_bottle ระบุค่าสี HSV ต่ำสุดที่ต้องการตรวจจับ (Hue: 95, Saturation: 80, Value: 100)
- upper_bottle ใช้ค่าสี HSV สูงสุด (Hue: 120, Saturation: 255, Value: 255)

ช่วงสีเหล่านี้ทำให้หุ่นยนต์สามารถมองเห็นฉลากขวดน้ำที่มีสีอยู่ในช่วงนี้ได้

การสร้าง Mask สำหรับแต่ละวัตถุ

สำหรับลูกไก่

```
mask_chicken = cv2.inRange(hsv_frame, lower_chicken, upper_chicken)
```

- cv2.inRange() : ฟังก์ชันนี้ใช้ในการสร้าง mask ซึ่งเป็นการกรอกภาพที่อยู่ในช่วงสีที่กำหนด หากสีในภาพ hsv_frame อยู่ในช่วงที่กำหนด lower_chicken ถึง upper_chicken จะได้ค่าสี 255 (สีขาวใน mask) แต่ถ้าสีไม่ได้อยู่ในช่วงที่กำหนดจะได้ค่าสีเป็น 0 (สีดำใน mask)

สำหรับขวดน้ำ (ฉลากขวดน้ำ)

```
mask_bottle = cv2.inRange(hsv_frame, lower_bottle, upper_bottle)
```

- กระบวนการเดียวกับ chicken แต่จะกรอกเฉพาะส่วนที่อยู่ในช่วง lower_bottle ถึง upper_bottle

รูปที่ 1 รูปภาพแสดงการอ่านภาพจากกล้อง (frame)

รูปที่ 2 รูปภาพแสดง mark หรือก็คือช่วงสีที่กำหนดของลูกไก่

รูปที่ 3 รูปภาพแสดง mark หรือก็คือช่วงสีที่กำหนดของฉลากขวดน้ำ

Contours

ผู้รับผิดชอบ: 6610110425 นายคีตศิลป์ คงสี

- ใช้ cv2.findContours() ในการตรวจหาขอบเขตของวัตถุในภาพ

```
contours_chicken, _ = cv2.findContours(mask_chicken, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours_bottle, _ = cv2.findContours(mask_bottle, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

  - การทำงาน:

    - mask_chicken และ mask_bottle: เป็นภาพ Mask ที่ได้จากการแปลงภาพต้นฉบับเป็นสี HSV แล้วทำการ threshold (ใช้ cv2.inRange()) เพื่อให้ภาพเป็นสีขาวดำ โดยส่วนที่ต้องการตรวจจับวัตถุ (ตุ๊กตาไก่หรือขวด) จะเป็นสีขาว (255) และส่วนที่เหลือจะเป็นสีดำ (0)
    - cv2.RETR_EXTERNAL: เป็นพารามิเตอร์ที่ใช้กำหนดวิธีการค้นหา contours โดย RETR_EXTERNAL จะดึงเฉพาะ contours ที่อยู่นอกสุดของวัตถุที่ถูกตรวจจับ (จะไม่สนใจขอบเขตภายในที่ซ้อนกัน)
    - cv2.CHAIN_APPROX_SIMPLE: เป็นพารามิเตอร์ที่ใช้กำหนดวิธีการเก็บข้อมูล contours โดย CHAIN_APPROX_SIMPLE จะลดจำนวนจุดที่ใช้เก็บ contours ให้น้อยลง ถ้าจุดเหล่านั้นสามารถแทนที่ด้วยเส้นตรง เช่น ในเส้นตรงจะเก็บแค่จุดหัวและจุดท้าย ไม่เก็บจุดทุกจุดในเส้นนั้น เพื่อประหยัดหน่วยความจำและประมวลผลได้เร็วขึ้น

  - ผลลัพธ์:

    - contours_chicken และ contours_bottle: เป็นลิสต์ของ contours ที่พบใน Mask นั้น ๆ โดยแต่ละ contour คือชุดของพิกัด (x, y) ที่เป็นขอบของวัตถุในภาพ
    - _ : ตัวแปรนี้ใช้แทนค่าที่ไม่ได้ใช้งาน ซึ่งในกรณีนี้จะเป็นข้อมูลลำดับชั้นของ contours ที่เราไม่ต้องการใช้งาน จึงใช้ _ เพื่อเก็บค่าและไม่ต้องสนใจ

  - สรุปการทำงาน:

    - cv2.findContours(mask_chicken, ...): ใช้เพื่อค้นหาขอบเขตของวัตถุ "ตุ๊กตาไก่" ในภาพ
    - cv2.findContours(mask_bottle, ...): ใช้เพื่อค้นหาขอบเขตของวัตถุ "ขวด" ในภาพ

รูปที่ 4 แสดงพิกัด (x, y) ที่ได้จาก cv2.findContours() ตำแหน่งA

รูปที่ 5 แสดงพิกัด (x, y) ที่ได้จาก cv2.findContours() ตำแหน่งB

รูปที่ 6 แสดงพิกัด (x, y) ที่ได้จาก cv2.findContours() ตำแหน่งC

- เลือก contours ที่ใหญ่ที่สุด

```
max_chicken_contour = max(contours_chicken, key=cv2.contourArea)
max_bottle_contour = max(contours_bottle, key=cv2.contourArea)
```

  - การทำงาน:

    - max(): หา contour ที่มีพื้นที่มากที่สุดในลิสต์ของ contours
    - key=cv2.contourArea: พารามิเตอร์นี้บอกให้ฟังก์ชัน max() ทำการเปรียบเทียบพื้นที่ของแต่ละ contour โดยใช้ฟังก์ชัน cv2.contourArea() เพื่อหาพื้นที่ของแต่ละ contour

  - ผลลัพธ์:

    - max_chicken_contour: จะเก็บ contour ของตุ๊กตาไก่ที่มีพื้นที่มากที่สุดจากลิสต์ contours_chicken
    - max_bottle_contour: จะเก็บ contour ของขวดที่มีพื้นที่มากที่สุดจากลิสต์ contours_bottle

  - สรุปการทำงาน:
    - คำสั่งนี้ใช้เพื่อเลือก ขอบเขตของวัตถุที่ใหญ่ที่สุด (ทั้งตุ๊กตาไก่และขวด) จากการค้นหาทั้งหมดในภาพ โดยอาศัยพื้นที่ของวัตถุเป็นเกณฑ์

รูปที่ 7 แสดงcontours ที่ใหญ่ที่สุดของตุ๊กตาไก่ และขวดน้ำ ตำแหน่งA

รูปที่ 8 แสดงcontours ที่ใหญ่ที่สุดของตุ๊กตาไก่ และขวดน้ำ ตำแหน่งB

รูปที่ 9 แสดงcontours ที่ใหญ่ที่สุดของตุ๊กตาไก่ และขวดน้ำ ตำแหน่งC

- ใช้ cv2.boundingRect() ในการหากรอบสี่เหลี่ยมรอบ contours ที่ใหญ่ที่สุด

```
x_chicken, y_chicken, w_chicken, h_chicken) = cv2.boundingRect(max_chicken_contour)
(x_bottle, y_bottle, w_bottle, h_bottle) = cv2.boundingRect(max_ bottle_contour)
```

  - การทำงาน:
    - cv2.boundingRect(contour): ใช้ในการคำนวณกรอบสี่เหลี่ยมผืนผ้าล้อมรอบ contour ที่ถูกส่งเข้ามา โดยฟังก์ชันจะคำนวณจากพิกัดขอบของ contour นั้น ๆ
    - ผลลัพธ์ที่ได้คือ (x, y, w, h):
      - x และ y คือพิกัดมุมซ้ายบนของกรอบสี่เหลี่ยมที่ล้อมรอบ contour
      - w คือความกว้างของกรอบสี่เหลี่ยม (width)
      - h คือความสูงของกรอบสี่เหลี่ยม (height)
    - สำหรับตุ๊กตาไก่:
      - max_chicken_contour เป็น contour ของตุ๊กตาไก่ที่มีพื้นที่มากที่สุด
      - ฟังก์ชัน cv2.boundingRect() จะคำนวณหากรอบสี่เหลี่ยมที่ครอบคลุม contour ของตุ๊กตาไก่ และส่งคืนค่าพิกัด (x, y) พร้อมกับขนาด (ความกว้าง w_chicken และความสูง h_chicken) ของกรอบนี้
    - สำหรับขวด:
      - max_bottle_contour เป็น contour ของขวดที่มีพื้นที่มากที่สุด
      - ฟังก์ชัน cv2.boundingRect() จะคำนวณหากรอบสี่เหลี่ยมที่ครอบคลุม contour ของขวด และส่งคืนค่าพิกัด (x, y) พร้อมกับขนาด (ความกว้าง w_bottle และความสูง h_bottle) ของกรอบนี้
  - ผลลัพธ์:
    - x_chicken, y_chicken, w_chicken, h_chicken: คือพิกัดและขนาดของกรอบสี่เหลี่ยมที่ล้อมรอบวัตถุ (ตุ๊กตาไก่)
    - x_bottle, y_bottle, w_bottle, h_bottle: คือพิกัดและขนาดของกรอบสี่เหลี่ยมที่ล้อมรอบวัตถุ (ขวด)
  - สรุปการทำงาน:
    - ฟังก์ชัน cv2.boundingRect() จะใช้ในการสร้างกรอบสี่เหลี่ยมผืนผ้ารอบ contour ที่เจอในภาพ โดยจะส่งคืนค่าพิกัดและขนาดของกรอบ

รูปที่ 10 แสดงกรอบสี่เหลี่ยมผืนผ้ารอบ contour ที่เจอในภาพ ตำแหน่ง A

รูปที่ 11 แสดงกรอบสี่เหลี่ยมผืนผ้ารอบ contour ที่เจอในภาพ ตำแหน่ง B

รูปที่ 12 แสดงกรอบสี่เหลี่ยมผืนผ้ารอบ contour ที่เจอในภาพ ตำแหน่ง C

ใช้ Opencv วาดกรอบสี่เหลี่ยมและใส่ข้อความ

ผู้รับผิดชอบ: 6610110341 นายสุธินันท์ รองพล

การวาดกรอบสี่เหลี่ยมของวัตถุ โดยมีวัตถุ 2 อัน คือ ตุ๊กตาไก่ และขวดน้ำ

1.กำหนดช่วงในการแสดงความสูงและความกว้างของสี่เหลี่ยมโดยกำหนดเงื่อนไขตามระยะและแบ่งตามตำแหน่งกระเบื้อง

แต่ละตำแหน่งของวัตถุบนกระเบื้องมีพื้นที่ contour ที่แตกต่างกัน โดยแบ่งเงื่อนไขตามความกว้างของวัตถุ (w) ซึ่งระยะทางระหว่างหุ่นยนต์กับวัตถุมีผลต่อขนาดของ contour ยิ่งหุ่นยนต์เข้าใกล้วัตถุมากเท่าไร พื้นที่ contour และความกว้างของวัตถุก็จะเพิ่มขึ้นตามไปด้วย

จากนั้นทำการปรับจูน ค่าความกว้าง(w) ความสูงของกรอบสี่เหลี่ยม(h) และตำแหน่งของแกน y เพื่อนำไปประกอบเป็นกรอบสี่เหลี่ยม

ตุ๊กตาไก่

```
if w_chick > 65:
    add_w_chick = int(w_chick * 0.3)
    add_h_chick = int(h_chick * 0.6)
    new_y_chick = y_chick - add_h_chick // 2 + 20
elif w_chick > 25:
    add_w_chick = int(w_chick * 0.35)
    add_h_chick = int(h_chick * 0.55)
    new_y_chick = (y_chick - add_h_chick // 2 + 20) - 15

else:
    add_w_chick = int(w_chick * 0.39)
    add_h_chick = int(h_chick * 0.59)
    new_y_chick = (y_chick - add_h_chick // 2 + 20) - 15
```

กำหนดเงื่อนไขเพื่อแบ่งช่วงความกว้างของวัตถุเพื่อตัดสินใจว่าจะขยายขนาดของกรอบและปรับตำแหน่งอย่างไร เพื่อล้อมรอบวัตถุ "chick" ได้อย่างเหมาะสมในแต่ละกรณี

ขวดน้ำ

```
if w_bottle > 85:
    add_w_bottle = int(w_bottle * 0.27)
    add_h_bottle = int(h_bottle * 2.7)
    new_y_bottle = y_bottle - add_h_bottle // 2 -13

elif w_bottle > 50:
    add_w_bottle = int(w_bottle * 0.27)
    add_h_bottle = int(h_bottle * 2.9)
    new_y_bottle = y_bottle - add_h_bottle // 2 -11

elif w_bottle > 35:
    add_w_bottle = int(w_bottle * 0.28)
    add_h_bottle = int(h_bottle * 3.3)
    new_y_bottle = y_bottle - add_h_bottle // 2 -4

else:
    add_w_bottle = int(w_bottle * 0.47)
    add_h_bottle = int(h_bottle * 3.4)
    new_y_bottle = y_bottle - add_h_bottle // 2 -4
```

กำหนดเงื่อนไขเพื่อแบ่งช่วงความกว้างของวัตถุโดยในที่นี้จะวัดบริเวณฉลากของขวดน้ำเพื่อตัดสินใจว่าจะขยายขนาดของกรอบและปรับตำแหน่งอย่างไร เพื่อล้อมรอบวัตถุ "bottle" ได้อย่างเหมาะสมในแต่ละกรณี ซึ่งในกรณีของขวดน้ำนั้นต้องคำนวณความสูงในส่วนที่เป็นสีใส

เมื่อปรับจูนค่าเสร็จสิ้น จะนำค่า w, h, y ที่ปรับจูนมาใช้ในการเลื่อนตำแหน่งของกรอบสี่เหลี่ยม

ตุ๊กตาไก่

```
new_x_chick = x_chick - add_w_chick // 2
new_w_chick = w_chick + add_w_chick
new_h_chick = h_chick + add_h_chick
```

ขวดน้ำ

```
new_x_bottle = x_bottle - add_w_bottle // 2
new_w_bottle = w_bottle + add_w_bottle
new_h_bottle = h_bottle + add_h_bottle
```

การวาดกรอบสี่เหลี่ยม

ตุ๊กตาไก่

```
cv2.rectangle(frame, (new_x_chick, new_y_chick),
              (new_x_chick + new_w_chick, new_y_chick + new_h_chick),
              (255, 0, 255), 2)
```

วาดกรอบสี่เหลี่ยมใหม่บนภาพ frame โดยใช้พิกัดเริ่มต้น new_x_chick, new_y_chick กำหนดสีของเส้นเป็นสีชมพูสีคู่ตรงข้ามของตุ๊กตาไก่ (RGB: 255, 0, 255) และความหนาเป็น 2 พิกเซล

ขวดน้ำ

```
cv2.rectangle(frame, (new_x_bottle, new_y_bottle),
              (new_x_bottle + new_w_bottle, new_y_bottle + new_h_bottle),
              (0, 165, 255), 2)
```

วาดกรอบสี่เหลี่ยมใหม่บนภาพ frame โดยใช้พิกัดเริ่มต้น new_x_bottle, new_y_bottle กำหนดสีของเส้นเป็นส้มสีคู่ตรงข้ามของ (RGB: 0, 165, 255) และความหนาเป็น 2 พิกเซล

การใส่ข้อความบน frame

ตุ๊กตาไก่

```
cv2.putText(frame, f" CHICK x: {x_chick},
        y: {y_chick},
        w: {w_chick},
        h: {h_chick}",
        (x_chick, y_chick - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
```

ขวดน้ำ

```
cv2.putText(frame, f" BOTTLE x: {x_bottle},
        y: {y_bottle},
        w: {w_bottle},
        h: {h_bottle}",
        (x_bottle, y_bottle-50),
        cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
```

ใช้ฟังก์ชัน cv2.putText เพื่อวางข้อความที่มีข้อมูลพิกัด (x, y), ความกว้าง (w), และความสูง (h) ของวัตถุลงบนภาพ และวางตำแหน่ง x, y – 50 (โดยเลื่อนขึ้นจากจุดเริ่มต้นของวัตถุ 50 พิกเซล) และใช้สีของข้อความเป็นสีขาว

ตำแหน่ง A

รูปที่ 13 ภาพจากกล้องและหุ่นยนต์ในตำแหน่ง A

ตำแหน่ง B

รูปที่ 14 ภาพจากกล้องและหุ่นยนต์ในตำแหน่ง B

ตำแหน่ง C

รูปที่ 15 ภาพจากกล้องและหุ่นยนต์ในตำแหน่ง C

แสดงผลภาพและการใช้ PID ควบคุมการเคลื่อนที่

ผู้รับผิดชอบ: 6610110327 นายสิรวิชญ์ น้อยผา

ในส่วนนี้จะเป็นการใช้ PID control เพื่อควบคุมการเคลื่อนที่ของหุ่นยนต์ให้ไปถึงเป้าหมายที่กำหนดไว้ โดยทำการคำนวณความผิดพลาดในแต่ละช่วงเวลาและปรับความเร็วของล้อหุ่นยนต์ตามค่าที่คำนวณได้

โดยตั้งเงื่อนไขการเคลื่อนที่ของหุ่นยนต์ คือ

กำหนดให้ ผลต่างของค่าของระยะทางเป้าหมายต้องกับตำแหน่งหุ่นยนต์ปัจจุบันต้องมากกว่าค่า tolerance หุ่นยนต์จึงจะเคลื่อนที่ โดยในการกำหนดระยะทางเป้าหมาย เนื่องจากวัตถุที่จะตรวจจับกับหุ่นยนต์ห่างกัน 2 กระเบื้อง และหุ่นยนต์ในตำแหน่ง D นั้นตั้งอยู่กึ่งกลางของกระเบื้อง ดังนั้น จึงกำหนดระยะทางเป้าหมายไว้ที่ 1.3 เมตร

```
if target_distance - current_x > tolerance:
```

ต่อมาในส่วนของการใช้ PID controller ในการปรับความเร็วการเคลื่อนที่ของหุ่นยนต์

ค่า error ถูกคำนวณโดยการหาความแตกต่างระหว่างตำแหน่งปัจจุบัน (current_x) และระยะห่างเป้าหมาย (target_distance)

```
error = target_distance - current_x
time_diff = current_time - prev_time
```

ในส่วนของ controller I

ค่า error สะสมถูกคำนวณโดยการสะสมค่าความผิดพลาดในแต่ละช่วงเวลาเพื่อแก้ไขข้อผิดพลาดระยะยาว

```
time_diff = current_time - prev_time
integral += error * time_diff
```

และในส่วนของ controller D

```
derivative = (error - prev_error) / time_diff if time_diff > 0 else 0.0
```

อัตราการเปลี่ยนแปลงของความผิดพลาด (derivative) คำนวณจากความแตกต่างของค่าความผิดพลาดระหว่างช่วงเวลา

ต่อมาในการนำ PID controller มาปรับ speed การเคลื่อนที่ของหุ่นยนต์

ค่าความเร็วของล้อหุ่นยนต์ (speed) ถูกคำนวณจากการรวมค่าของ Proportional (P), Integral (I), และ Derivative (D) จากนั้นจะทำการจำกัดค่าความเร็วให้อยู่ระหว่าง 0 ถึง 20 เพื่อให้แน่ใจว่าความเร็วไม่สูงเกินไปหรือต่ำเกินไป

```
speed = kp * error + kd * derivative + ki * integral
speed = max(min(speed, 20), 0)
```

จากนั้นนำ ค่า speed ที่ได้มาใช้ในการออกคำสั่งการเคลื่อนที่ขอหุ่นยนต์

```
ep_chassis.drive_wheels(w1=speed, w2=speed, w3=speed, w4=speed)
```

อัพเดทค่า error และตำแหน่งของหุ่นยนต์ เพื่อใช้ในการคำนวณรอบต่อไป

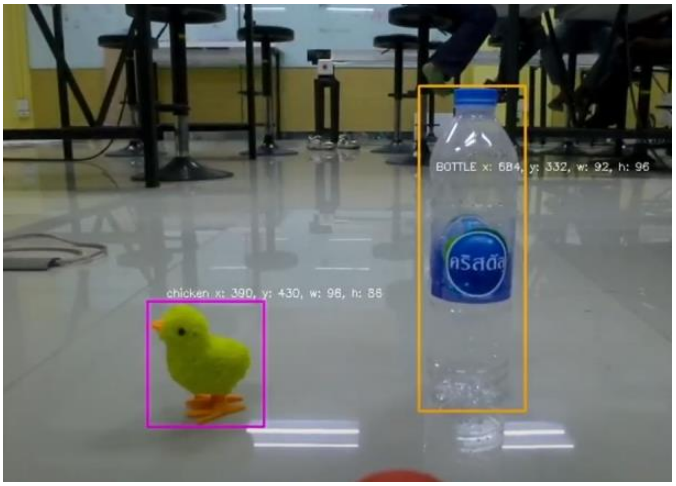
```
prev_error = error
prev_time = current_time
```

เมื่อหุ่นยนต์ทำงานจนเสร็จสิ้น หรือทำงานไม่ตรงตามเงื่อนไขก็ให้หุ่นยนต์หยุดการเคลื่อนที่

```
else:
    ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
```

โดยในการใช้ PID controller ในการปรับ speed การเคลื่อนที่ของหุ่นยนต์จะทำให้การแสดงผลภาพการตรวจจับวัตถุของหุ่นยนต์ robomaster นั่นมีความเสถียรมากยิ่งขึ้น จะช่วยทำให้ตัวหุ่นเคลื่อนที่ได้นิ่ง ไม่เคลื่อนที่เร็วเกินไปจนทำให้ตัวหุ่นยนต์อาจจะเกิดการชนวัตถุที่ตรวจจับ หรืออาจทำให้การเปลี่ยนเฟรมของภาพอาจเกิดขึ้นอย่างรวดเร็ว ทำให้กล้องจับภาพได้ไม่เต็มที่และทำให้การหาขอบเขตของวัตถุทำได้ยากขึ้น

รูปที่ 16 : ภาพแสดงผลลัพธ์การเคลื่อนที่ของหุ่นยนต์ ที่หยุดนิ่งเมื่อถึงระยะทางเป้าหมาย



รูปที่ 17 : ภาพแสดงผลลัพธ์การตรวจจับวัตถุ เมื่อหุ่นยนต์หยุดนิ่ง

Code

```
import cv2

import robomaster

from robomaster import robot, camera

import time

import numpy as np



def sub_position_handler(position_info):

    global current_x

    current_x, y, z = position_info



def detect_chicken():

    max_chicken_contour = max(contours_chicken, key=cv2.contourArea)

    (x_chicken, y_chicken, w_chicken, h_chicken) = cv2.boundingRect(max_chicken_contour)



    if w_chicken > 65:

        add_w_chicken = int(w_chicken * 0.3)

        add_h_chicken = int(h_chicken * 0.6)

        new_y_chicken = y_chicken - add_h_chicken // 2 + 20



    elif w_chicken > 25:

        add_w_chicken = int(w_chicken * 0.35)

        add_h_chicken = int(h_chicken * 0.55)
```

```
        new_y_chicken = (y_chicken - add_h_chicken // 2 + 20) - 15


    else:
        add_w_chicken = int(w_chicken * 0.39)
        add_h_chicken = int(h_chicken * 0.59)
        new_y_chicken = (y_chicken - add_h_chicken // 2 + 20) - 15


    new_x_chicken = x_chicken - add_w_chicken // 2
    new_w_chicken = w_chicken + add_w_chicken
    new_h_chicken = h_chicken + add_h_chicken


    cv2.rectangle(frame, (new_x_chicken, new_y_chicken), (new_x_chicken + new_w_chicken, new_y_chicken +
new_h_chicken), (255, 0, 255), 2)
    cv2.putText(frame, f" chicken x: {x_chicken}, y: {y_chicken}, w: {w_chicken}, h: {h_chicken}", (x_chicken,
y_chicken - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)


def detect_bottle():
    max_bottle_contour = max(contours_bottle, key=cv2.contourArea)
    (x_bottle, y_bottle, w_bottle, h_bottle) = cv2.boundingRect(max_bottle_contour)


    if w_bottle > 85:
        add_w_bottle = int(w_bottle * 0.27)
        add_h_bottle = int(h_bottle * 2.7)
        new_y_bottle = y_bottle - add_h_bottle // 2 -13
```

```
    elif w_bottle > 50:
        add_w_bottle = int(w_bottle * 0.27)
        add_h_bottle = int(h_bottle * 2.9)
        new_y_bottle = y_bottle - add_h_bottle // 2 -11

    elif w_bottle > 35:
        add_w_bottle = int(w_bottle * 0.28)
        add_h_bottle = int(h_bottle * 3.3)
        new_y_bottle = y_bottle - add_h_bottle // 2 -4

    else:
        add_w_bottle = int(w_bottle * 0.47)
        add_h_bottle = int(h_bottle * 3.4)
        new_y_bottle = y_bottle - add_h_bottle // 2 -4

    new_x_bottle = x_bottle - add_w_bottle // 2
    new_w_bottle = w_bottle + add_w_bottle
    new_h_bottle = h_bottle + add_h_bottle

    cv2.rectangle(frame, (new_x_bottle, new_y_bottle), (new_x_bottle + new_w_bottle, new_y_bottle + new_h_bottle), (0, 165, 255), 2)
    cv2.putText(frame, f" BOTTLE x: {x_bottle}, y: {y_bottle}, w: {w_bottle}, h: {h_bottle}", (x_bottle, y_bottle-50), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
```

```
if __name__ == "__main__":
    ep_robot = robot.Robot()
    ep_robot.initialize(conn_type="ap")
    ep_camera = ep_robot.camera
    ep_gimbal = ep_robot.gimbal
    ep_chassis = ep_robot.chassis

    center_x = 1280 / 2
    center_y = 720 / 2

    current_x = 0.0
    target_distance = 1.3
    kp, ki, kd = 75, 10, 30
    tolerance = 0.01
    prev_error, integral = 0.0, 0.0
    prev_time = time.time()

    ep_chassis.sub_position(freq=10, callback=sub_position_handler)
    ep_camera.start_video_stream(display=False, resolution=camera.STREAM_720P)
    ep_gimbal.recenter(pitch_speed=200, yaw_speed=200).wait_for_completed()
    ep_gimbal.moveto(pitch=-10, yaw=0).wait_for_completed()

    while True:
```

```
frame = ep_camera.read_cv2_image(strategy="newest", timeout=0.5)


hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)


lower_chicken = np.array([33, 150, 100])

upper_chicken = np.array([38, 255, 255])


lower_bottle = np.array([95, 80, 100])

upper_bottle = np.array([120, 255, 255])


mask_chicken = cv2.inRange(hsv_frame, lower_chicken, upper_chicken)

mask_bottle = cv2.inRange(hsv_frame, lower_bottle, upper_bottle)


contours_chicken, _ = cv2.findContours(mask_chicken, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

contours_bottle, _ = cv2.findContours(mask_bottle, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)


if contours_chicken: detect_chicken()

if contours_bottle: detect_bottle()


if target_distance - current_x > tolerance :

    current_time = time.time()

    error = target_distance - current_x

    time_diff = current_time - prev_time

    integral += error * time_diff
```

```
            derivative = (error - prev_error) / time_diff if time_diff > 0 else 0.0
            speed = kp * error + kd * derivative + ki * integral
            speed = max(min(speed, 20), 0)
            ep_chassis.drive_wheels(w1=speed, w2=speed, w3=speed, w4=speed)
            prev_error = error
            prev_time = current_time
            time.sleep(0.05)

        else:
            ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
            time.sleep(0.05)

        cv2.imshow("Frame", frame)
        if cv2.waitKey(1) & 0xFF == ord("q"): break

        time.sleep(0.1)

    cv2.destroyAllWindows()
    ep_camera.stop_video_stream()
    ep_chassis.unsub_position()
    ep_robot.close()
```

ปัญหาที่พบเจอ

- ตรวจจับวัตถุอื่นที่มีค่าสีอยู่ในช่วงเดียวกัน และมีขนาดใหญ่กว่า

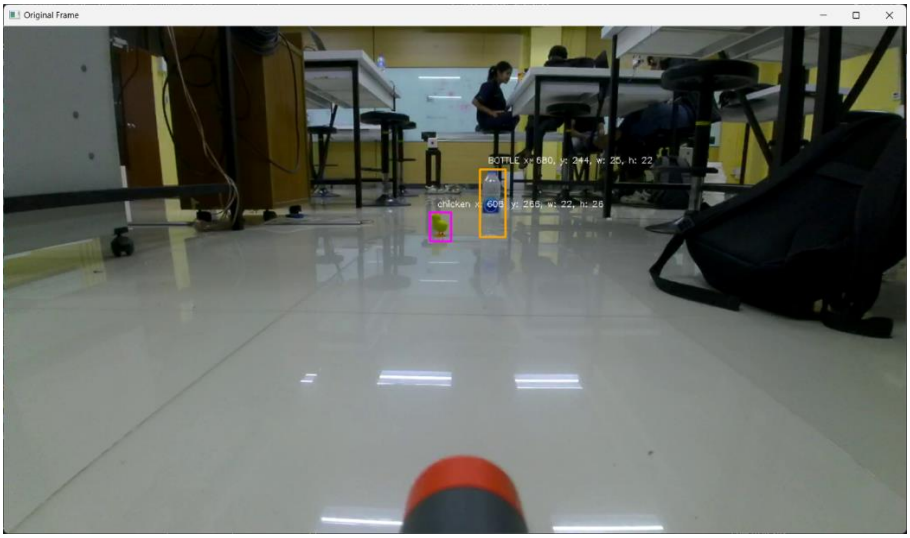
แนวทางแก้ไข

- **ครอบภาพ :**
  - กำหนดพื้นที่เฉพาะในภาพที่คาดว่าวัตถุเป้าหมายจะปรากฏ จากนั้นจึงตรวจจับcontour เฉพาะในบริเวณที่ครอบภาพไว้เท่านั้น เพื่อลดการตรวจจับวัตถุอื่นที่อยู่นอกขอบเขตนี้
- **ปรับช่วงสี :**
  - ปรับปรุงช่วงของค่าสีใน HSV space ให้ละเอียดและเหมาะสมมากขึ้นเพื่อให้แยกความแตกต่างระหว่างวัตถุเป้าหมายกับวัตถุที่ไม่ต้องการตรวจจับ เช่น ลองปรับค่า hue, saturation, และ value ให้แคบลง
- **ตรวจสอบรูปร่าง :**
  - ใช้การตรวจสอบรูปร่างของcontour เช่น การคำนวณสัดส่วนระหว่างความกว้างและความสูง เพื่อกรองเฉพาะวัตถุที่มีรูปร่างใกล้เคียงกับที่ต้องการ

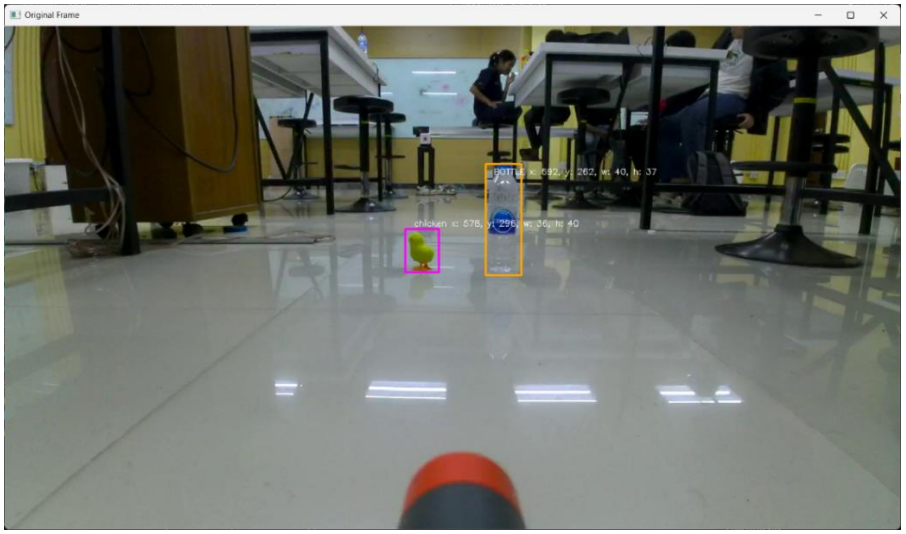
ผลลัพธ์ และ Video ตอนฝึกซ้อม 2 วิดิโอ

https://youtu.be/pxKeGGdQurw?si=rw7lG1jZi3VXOAbS

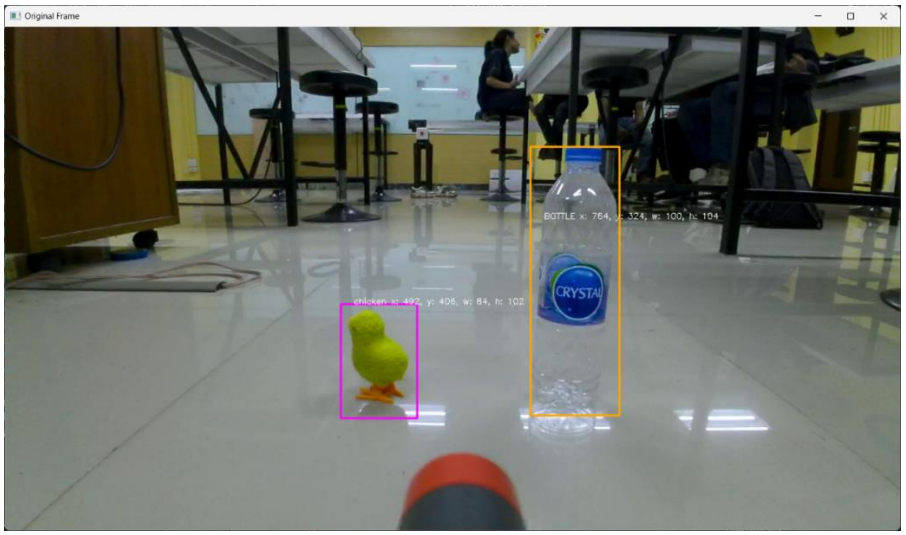
https://youtu.be/yG_v9grZpBE?si=onuJJK6hrFUNJqOX



รูปที่ 18 ผลลัพธ์: สร้างกรอบสี่เหลี่ยมผืนผ้ารอบตุ๊กตาไก่ และขวดน้ำ ตำแหน่ง A



รูปที่ 19 ผลลัพธ์: สร้างกรอบสี่เหลี่ยมผืนผ้ารอบตุ๊กตาไก่ และขวดน้ำ ตำแหน่ง B



รูปที่ 20 ผลลัพธ์: สร้างกรอบสี่เหลี่ยมผืนผ้ารอบตุ๊กตาไก่ และขวดน้ำ ตำแหน่ง C