Human Body Following

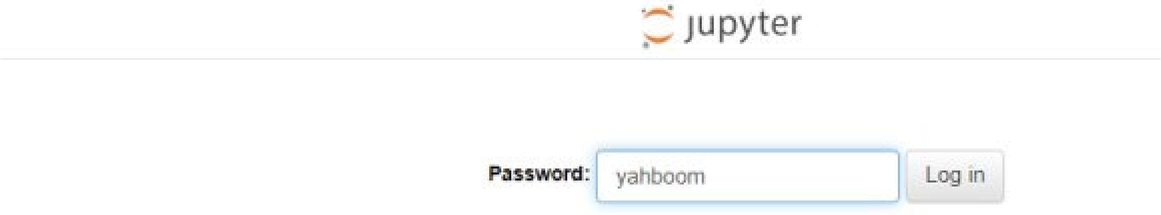
This tutorial is specific to the CM5 version and will not work directly with the CM4 version

1. Experimental Purpose

Learn to use the robot dog's camera to locate and follow a human body.

2. Experimental Steps

Log in to the robot dog's system, exit the robot dog program, and enter "ip (where ip is the robot dog's IP address):8888 in your browser. Once logged in, enter the password "yahboom"

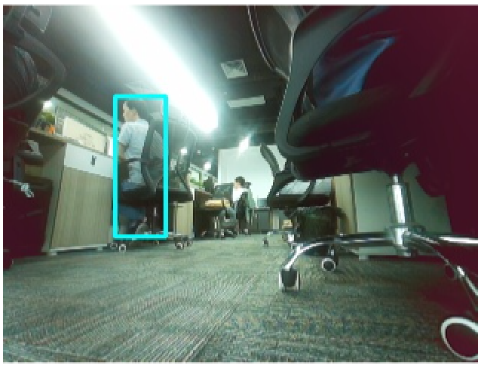


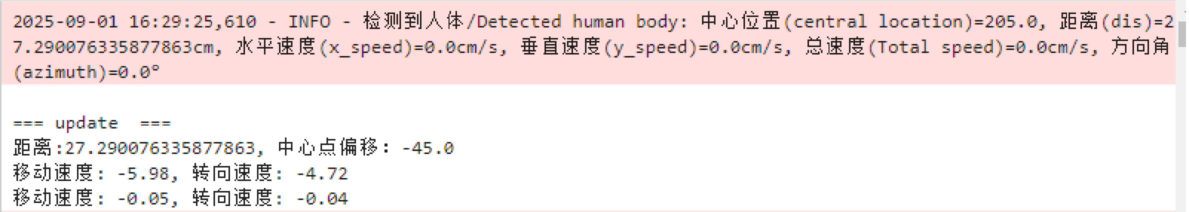
Then log in.
Go to **cd ~/DOGZILLA_Lite_class/6.AI Visual Interaction Course/13.Human Body Follows** and run **follow_person.ipynb**.

Finally, run the program.

3. Experimental Results

After running the program, the robot dog will detect a human and follow the human body by moving forward, backward, left, or right, depending on the distance from the human body.





```
2025-09-01 16:29:25,610 - INFO - 检测到人体/Detected human body: 中心位置(central location)=205.0, 距离(dis)=2
7.290076335877863cm, 水平速度(x_speed)=0.0cm/s, 垂直速度(y_speed)=0.0cm/s, 总速度(Total speed)=0.0cm/s, 方向角
(azimuth)=0.0°

=== update  ===
距离:27.290076335877863, 中心点偏移: -45.0
移动速度: -5.98, 转向速度: -4.72
移动速度: -0.05, 转向速度: -0.04
```

4. Experimental Source Code Analysis

```
# Display the output area
display(image_widget)
display(output_area)

from track import HumanTracker
from xgolib import XGO

button = Button()
dog = XGO(port="/dev/ttyAMA0", version="xgolite")
dog.attitude('p', -10)


tracker = HumanTracker()
from PIL import Image, ImageDraw
import xgoscreen.LCD_2inch as LCD_2inch
splash_theme_color = (255,255,255)
mydisplay = LCD_2inch.LCD_2inch()
mydisplay.Init()
mydisplay.clear()
# Init Splash
splash = Image.new("RGB", (mydisplay.height, mydisplay.width),
splash_theme_color)
draw = ImageDraw.Draw(splash)
mydisplay.ShowImage(splash)
detector = HumanDetector()
last_move_x_speed = 0.0
last_turn_speed = 0.0
filter_coefficient = 0.3  # 滤波系数 filter coefficient
while True:
    with output_area:
        frame = picam2.capture_array()
        humans = detector.detect_humans(frame)
        for human in humans:
            if human:
                print("\n=== update  ===")
                center_x, distance = human['center_x'], human['distance'] if
human else (None, None)
```

```
        #print(f"距离最近的人在 ({human['bbox']}), center_x:
{human['center_x']}, distance: {human['distance']} cm")
        tracking_params = tracker.update(human)
        if tracking_params['is_detected'] and
tracking_params['is_moving']:
          prediction =
tracker.predict_future_position(tracking_params, 0.06)
          if prediction:
            pass
        raw_move_x_speed = tracking_params['velocity_x'] * 0.05 +
(distance - 100) * 0.1
        if -20 <distance - 70 <  20:
          raw_move_x_speed = (distance - 70) * 0.3
        else:
          raw_move_x_speed = (distance - 70) * 0.2
        raw_turn_speed = tracking_params['velocity_x'] * 0.02
        if 145<center_x<175:
          raw_turn_speed = 0
        elif 130<center_x<145 or 175< center_x <190:
          raw_turn_speed = (160 - center_x) * 0.2
        else:
          raw_turn_speed = (160 - center_x) * 0.15
        move_x_speed = filter_coefficient * last_move_x_speed + (1 -
filter_coefficient) * raw_move_x_speed
        turn_speed = filter_coefficient * last_turn_speed + (1 -
filter_coefficient) * raw_turn_speed
        max_move_speed = 10.0
        max_turn_speed = 15.0
        move_x_speed = max(-max_move_speed, min(max_move_speed,
move_x_speed))
        turn_speed = max(-max_turn_speed, min(max_turn_speed,
turn_speed))
        # 更新上一时刻的速度 Update the speed from the previous moment
        last_move_x_speed = move_x_speed
        last_turn_speed = turn_speed
        if la =="cn":
          print(f'距离:{distance}, 中心点偏移: {160-center_x}')
          print(f"移动速度: {move_x_speed:.2f}, 转向速度:
{turn_speed:.2f}")
        else:
          print(f'distance:{distance}, center_x: {160-center_x}')
          print(f"move_x_speed: {move_x_speed:.2f}, turn_speed:
{turn_speed:.2f}")

        #dog.move_x(move_x_speed)
        #dog.turn(turn_speed)
        # #dog.move_x(0.02 * tracking_params['velocity_y'])
        # #dog.turn(tracking_params['velocity_x'])


    # 如果没有检测到人体，逐渐减速 If no human body is detected, gradually slow
down
    if humans == []:
      move_x_speed = 0.03*filter_coefficient * last_move_x_speed
      turn_speed = 0.03*filter_coefficient * last_turn_speed
      max_move_speed = 3.0
      max_turn_speed = 3.0
```

```
            move_x_speed = max(-max_move_speed, min(max_move_speed, move_x_speed))
            turn_speed = max(-max_turn_speed, min(max_turn_speed, turn_speed))
            dog.move_x(move_x_speed)
            dog.turn(turn_speed)

            if la =="cn":
                print(f"移动速度: {move_x_speed:.2f}, 转向速度: {turn_speed:.2f}")
            else:
                print(f"move_x_speed: {move_x_speed:.2f}, turn_speed: {turn_speed:.2f}")

            last_move_x_speed = move_x_speed
            last_turn_speed = turn_speed

        b, g, r = cv2.split(frame)
        image_widget.value = bgr8_to_jpeg(frame)
        #cv2.imshow('Human Detection', frame)

        img = cv2.merge((r, g, b))
        imgok = Image.fromarray(img)
        mydisplay.ShowImage(imgok)

        if button.press_b():
            dog.reset()
            break
```

```
if -20 <distance - 70 <  20:
    raw_move_x_speed = (distance - 70) * 0.3
else:
    raw_move_x_speed = (distance - 70) * 0.2
raw_turn_speed = tracking_params['velocity_x'] * 0.02
if 145<center_x<175:
    raw_turn_speed = 0
elif 130<center_x<145 or 175< center_x <190:
    raw_turn_speed = (160 - center_x) * 0.2
else:
    raw_turn_speed = (160 - center_x) * 0.15
```

- If you want to change the following distance and the left and right turn range, you can modify the range here.
- detector.detect_humans(frame): This function detects the position of human bodies in the image.