Palm Targeting

1. Content Description

This course implements color image acquisition and palm detection using the MediaPipe framework, outputting the palm coordinates. This can then be combined with a robot chassis or robotic arm to track hand movements. This section requires entering commands in a terminal. The terminal you open depends on your motherboard type. This course uses a Raspberry Pi 5 as an example.

For Raspberry Pi and Jetson Nano boards, you need to open a terminal on the host computer and enter the command to enter a Docker container. Once inside the Docker container, enter the commands mentioned in this section in the terminal. For instructions on entering a Docker container from the host computer, refer to **[01. Robot Configuration and Operation Guide] -- [5.Enter Docker (For JETSON Nano and RPi 5)]**.

For Orin boards, simply open a terminal and enter the commands mentioned in this section.

2. Program Startup

For the Raspberry Pi 5 controller, you must first enter the Docker container. For the Orin controller, this is not necessary.

Enter the Docker container (for steps, see [Docker Course] --- [4. Docker Startup Script]).

All of the following Docker commands must be executed from the same Docker container **(for steps, see [Docker Course] --- [3. Docker Submission and Multi-Terminal Access]).**

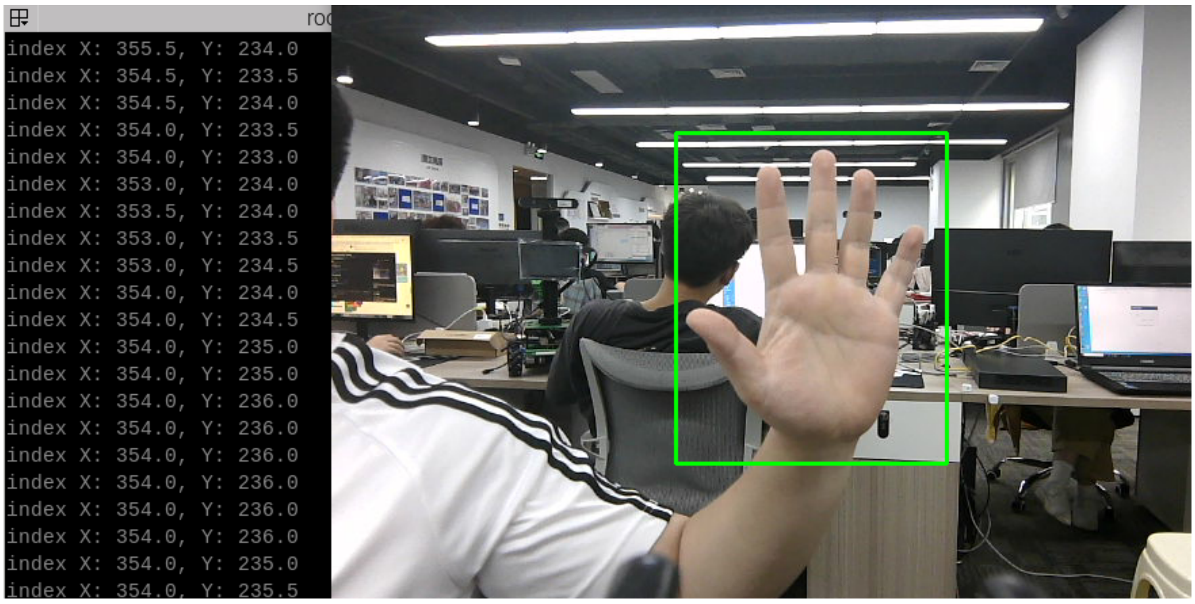
First, enter the following command in the terminal to start the camera.

```
#usb camera
ros2 launch usb_cam camera.launch.py
#nuwa camera
ros2 launch ascamera hp60c.launch.py
```

After successfully starting the camera, open another terminal and enter the following command to start the palm tracking program.

```
ros2 run yahboomcar_mediapipe 11_FindHand
```

After starting the program, as shown in the image below, when a hand is detected, it will be outlined in green on the screen, and the center coordinates of the hand will be output in the terminal.



```
index X: 355.5, Y: 234.0
index X: 354.5, Y: 233.5
index X: 354.5, Y: 234.0
index X: 354.0, Y: 233.5
index X: 354.0, Y: 233.0
index X: 353.0, Y: 234.0
index X: 353.5, Y: 234.0
index X: 353.0, Y: 233.5
index X: 353.0, Y: 234.5
index X: 354.0, Y: 234.0
index X: 354.0, Y: 234.5
index X: 354.0, Y: 235.0
index X: 354.0, Y: 235.0
index X: 354.0, Y: 236.0
index X: 354.0, Y: 236.0
index X: 354.0, Y: 236.0
index X: 354.0, Y: 236.0
index X: 354.0, Y: 236.0
index X: 354.0, Y: 236.0
index X: 354.0, Y: 235.0
index X: 354.0, Y: 235.5
```

3. Core Code Analysis

Program Code Path:

- Raspberry Pi 5 and Jetson Nano Board

The program code is running in Docker. The path in Docker is `/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/11_FindHand.py`

- Orin motherboard

The program code path is `/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/11_FindHand.py`

Import the library files used.

```python
import threading
import numpy as np
import time
import os
import sys
import cv2 as cv
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from std_msgs.msg import Bool
from geometry_msgs.msg import Twist
from cv_bridge import CvBridge
sys.path.append('/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_me
diapipe/yahboomcar_mediapipe')
from media_library import *
```

Initialize data and define publishers and subscribers,

```python
ef __init__(self):
    super().__init__('hand_ctrl_arm_node')
    #Call the media_library library to create an object of the HandDetector
class
```

```
        self.hand_detector = HandDetector()
        #create a publisher
        self.bridge = CvBridge()
        #Define subscribers for the color image topic
        camera_type = os.getenv('CAMERA_TYPE', 'usb')
        topic_name = '/ascamera_hp60c/camera_publisher/rgb0/image' if camera_type ==
'nuwa' else '/usb_cam/image_raw'

        self.subscription = self.create_subscription(
            Image,
            topic_name,
            self.image_callback,
            10)
```

Color image callback function,

```
def image_callback(self, msg):
    #Use CvBridge to convert color image message data into image data
    frame = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
    #Pass the obtained image into the process function for palm detection
    frame = self.process(frame)
    cv.imshow('frame', frame)
```

`process` function,

```
def process(self, frame):
    # Call the object method to perform palm detection and return the detected
image as well as the lmList list and bbox list
    frame, lmList, bbox = self.hand_detector.findHands(frame)
    # If the lmList list is not empty, it means that a palm has been detected.
    if len(lmList) != 0:
        threading.Thread(target=self.find_hand_threading, args=(lmList,
bbox)).start()
    return frame
```

`find_hand_threading` function,

```
def find_hand_threading(self, lmList, bbox):
    fingers = self.hand_detector.fingersUp(lmList)
    angle = self.hand_detector.ThumbTOforefinger(lmList)
    value = np.interp(angle, [0, 70], [185, 20])
    # Get the xy coordinates of the palm. bbox stores the xy coordinates of the
upper left and lower right corners of the palm
    indexX = (bbox[0] + bbox[2]) / 2
    indexY = (bbox[1] + bbox[3]) / 2
    print("index X: %.1f, Y: %.1f" % (indexX, indexY))
```

The definition of the Medipipe recognition class can be found in the media_library library, located in the yahboomcar_mediapipe package.

- Raspberry Pi 5 and Jetson-Nano boards

  The path in Docker is `/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/media_library.py`

- Orin boards

  The program code path is `/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/media_library.py`

In this library, we leverage the native mediapipe library to extend and define multiple classes. Each class defines different functions.

We simply pass in parameters as needed. For example, the HandDetector class defines the following functions:

- findHands: Finds hands
- fingersUp: Fingers extended upward
- ThumbTOforefinger: Detects the angle between the thumb and index finger
- get_gesture: Detects gestures