3D object recognition

1. Content Description

This course implements color image acquisition and 3D object recognition using the MediaPipe framework (only four examples are shown: a mug, a shoe, a chair, and a camera). This section requires entering commands in a terminal. The terminal you open depends on your board type. This lesson uses the Raspberry Pi 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 the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container from the host computer, refer to **[01. Robot Configuration and Operation Guide] --[4.Enter Docker (For JETSON Nano and RPi 5)]**.

For RDKX5 and Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

2. Program Startup

For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

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

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

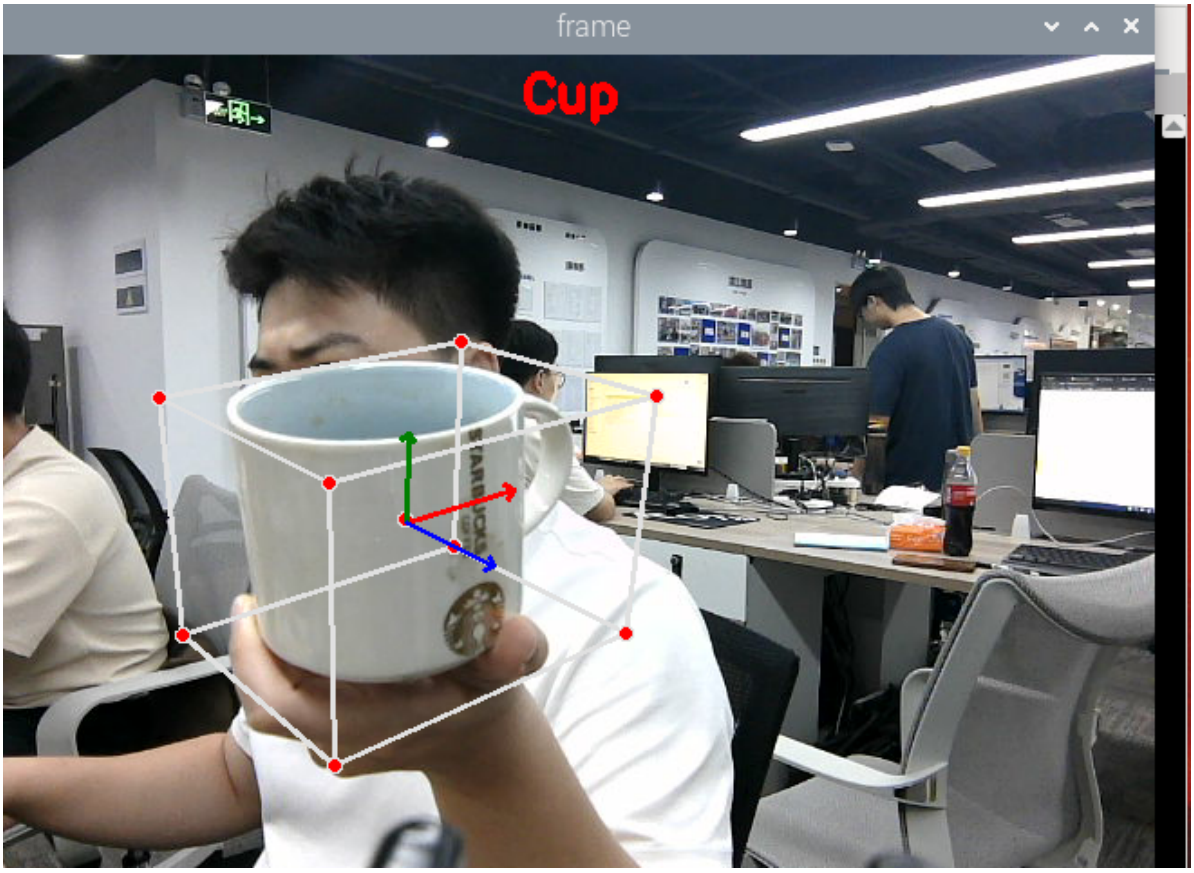
First, enter the following command in the terminal to start the camera.

```
#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 3D object detection program.

```
ros2 run yahboomcar_mediapipe 07_Objectron
```

After running the program, the default object to be recognized is shoes. You can press the F key to switch the object to be recognized. As shown in the image below, a mug is recognized.



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/07_Objectron.py`

- Orin board

  The program code path is `/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/07_Objectron.py`

- RDK X5

  The program code path is `/home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/07_Objectron.py`

Import the library files used.

```
import mediapipe as mp
import cv2 as cv
import time
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import os
import numpy as np
```

Initialize data and define publishers and subscribers,

```
def __init__(self, staticMode=False, maxObjects=5, minDetectionCon=0.5,
minTrackingCon=0.99):
    super().__init__('objectron')
    self.staticMode = staticMode
    self.maxObjects = maxObjects
    self.minDetectionCon = minDetectionCon
    self.minTrackingCon = minTrackingCon
    self.index = 0
    self.modelNames = ['Shoe', 'Chair', 'Cup', 'Camera']
    #Use the class in the mediapipe library to define a 3D detection object
    self.mpObjectron = mp.solutions.objectron
    self.mpDraw = mp.solutions.drawing_utils
    self.mpobjectron = self.mpObjectron.Objectron(
    self.staticMode, self.maxObjects, self.minDetectionCon, self.minTrackingCon,
self.modelNames[self.index])
    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):
    frame = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
    action = cv2.waitKey(1)
    #Press the F key to switch the recognized objects
    if action == ord('f') or action == ord('F') : self.configUP()
    frame = self.findObjectron(frame)
    cv.imshow('Objectron', frame)
```

configUP switches the object recognition function and selects self.modelNames by modifying the value of self.index. The options for self.modelNames are ['Shoe', 'Chair', 'Cup', 'Camera']

```
def configUP(self):
    self.index += 1
    if self.index>=4:self.index=0
        self.mpobjectron = self.mpObjectron.Objectron(
        self.staticMode, self.maxObjects, self.minDetectionCon,
self.minTrackingCon,            self.modelNames[self.index])
```

findObjectron object recognition function,

```
def findObjectron(self, frame):
    cv.putText(frame, self.modelNames[self.index], (int(frame.shape[1] / 2) -
30, 30),
    cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 3)
```

```
    #Convert the color space of the incoming image from BGR to RGB to facilitate
subsequent image processing
    img_RGB = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    #Call the process function in the mediapipe library to process the image.
During init, the self.pose object is created and initialized.
    results = self.mpobjectron.process(img_RGB)
    #Determine whether the target object is recognized
    if results.detected_objects:
        for id, detection in enumerate(results.detected_objects):
            #On the image, draw the coordinate system for the identified target
object
            self.mpDraw.draw_landmarks(frame, detection.landmarks_2d,
self.mpObjectron.BOX_CONNECTIONS)
            self.mpDraw.draw_axis(frame, detection.rotation,
detection.translation)
    return frame
```