

Multimodal visual understand + PTZ tracking(Voice Version)

Multimodal visual understand + PTZ tracking(Voice Version)

1. Course Content
2. Preparation
 - 2.1 Content Description
3. Running the Example
 - 3.1 Starting the Program
 - 3.2 Test Case
 - 3.2.1 Example 1: "Start Tracking Red"
 - 3.2.2 Example 2: "Please track the object in my hand"
4. Source Code Analysis
 - 4.1 Example 1
 - 4.2 Example 2

1. Course Content

1. Learn to use the robot's visual understanding capabilities
2. Learn new key source code

2. Preparation

2.1 Content Description

This lesson uses Jetson Orin NANO as an example. For users of the gimbal servo USB camera version, this is used 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.

💡 This example uses `model:"qwen/qwen2.5-v1-72b-instruct:free", "qwen-v1-latest"`

⚠️ The responses from the big model for the same test command may not be exactly the same and may differ slightly from the screenshots in the tutorial. If you need to increase or decrease the diversity of the big model's responses, refer to the section on configuring the decision-making big model parameters in the **[03.AI Model Basics] -- [5.Configure AI large model]**.

⚡ It is recommended that you first try the previous visual example. This example adds voice functionality to the singleton, and while the functionality is largely the same, I will not elaborate on the program implementation, code debugging, or results in detail!!!

3. Running the Example

3.1 Starting the Program

For Raspberry Pi P15 and jetson nano, you need to enter the Docker container first. For RDKX5 and Orin main controllers, this is not necessary.

Open a terminal on the vehicle and enter the following command:

```
ros2 launch largemode1 largemode1_control.launch.py
```

```
jetson@yahboom: ~ 115x33
IP_Address_1: 192.168.11.198
IP_Address_2: 192.168.11.195
-----
ROS_DOMAIN_ID: 62 | ROS: humble
my_robot_type: A1 | my_lidar: c1 | my_camera: usb
-----
jetson@yahboom:~$ ros2 launch largemode1 largemode1_control.launch.py
[INFO] [launch]: All log files can be found below /home/jetson/.ros/log/2025-08-20-15-55-46-177675-yahboom-56818
[INFO] [launch]: Default logging verbosity is set to INFO
----- robot_type = A1, rplidar_type = c1, camera_type = usb -----
----- robot_type = A1 -----
[INFO] [usb_cam_node_exe-1]: process started with pid [56875]
[INFO] [joint_state_publisher-2]: process started with pid [56877]
[INFO] [robot_state_publisher-3]: process started with pid [56879]
[INFO] [Ackman_driver_A1-4]: process started with pid [56881]
[INFO] [base_node_A1-5]: process started with pid [56883]
[INFO] [imu_filter_madgwick_node-6]: process started with pid [56885]
[INFO] [ekf_node-7]: process started with pid [56887]
```

After initialization is complete, the following content will be displayed.

```
jetson@yahboom: ~ 119x33
[usb_cam_node_exe-1] [INFO] [1755676548.086026008] [usb_cam]: camera calibration URL: package://usb_cam/config/camera_info.yaml
[usb_cam_node_exe-1] [INFO] [1755676548.165251803] [usb_cam]: Starting 'test_camera' (/dev/video0) at 640x480 via mmap (mjpeg2rgb) at 120 FPS
[usb_cam_node_exe-1] [swscaler @ 0xaab21f59650] No accelerated colorspace conversion found from yuv422p to rgb24.
[usb_cam_node_exe-1] This device supports the following formats:
[usb_cam_node_exe-1] Motion-JPEG 1280 x 720 (60 Hz)
[usb_cam_node_exe-1] Motion-JPEG 1920 x 1080 (30 Hz)
[usb_cam_node_exe-1] Motion-JPEG 1024 x 768 (30 Hz)
[usb_cam_node_exe-1] Motion-JPEG 640 x 480 (120 Hz)
[usb_cam_node_exe-1] Motion-JPEG 800 x 600 (60 Hz)
[usb_cam_node_exe-1] Motion-JPEG 1280 x 1024 (30 Hz)
[usb_cam_node_exe-1] Motion-JPEG 320 x 240 (120 Hz)
[usb_cam_node_exe-1] YUYV 4:2:2 1280 x 720 (9 Hz)
[usb_cam_node_exe-1] YUYV 4:2:2 1920 x 1080 (6 Hz)
[usb_cam_node_exe-1] YUYV 4:2:2 1024 x 768 (6 Hz)
[usb_cam_node_exe-1] YUYV 4:2:2 640 x 480 (30 Hz)
[usb_cam_node_exe-1] YUYV 4:2:2 800 x 600 (20 Hz)
[usb_cam_node_exe-1] YUYV 4:2:2 1280 x 1024 (6 Hz)
[usb_cam_node_exe-1] YUYV 4:2:2 320 x 240 (30 Hz)
[joint_state_publisher-2] [INFO] [1755676548.190477321] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
[usb_cam_node_exe-1] [INFO] [1755676548.970677305] [usb_cam]: Setting 'white_balance_temperature_auto' to 1
[usb_cam_node_exe-1] [INFO] [1755676548.970799444] [usb_cam]: Setting 'exposure_auto' to 1
[usb_cam_node_exe-1] [INFO] [1755676548.970816792] [usb_cam]: Setting 'exposure' to 150
[usb_cam_node_exe-1] [INFO] [1755676549.029402676] [usb_cam]: Setting 'focus_auto' to 0
[usb_cam_node_exe-1] [INFO] [1755676549.104723259] [usb_cam]: Timer triggering every 8 ms
[imu_filter_madgwick_node-6] [INFO] [1755676550.671276315] [imu_filter_madgwick]: First IMU message received.
[asr-14] [INFO] [1755676559.939338782] [asr]: The online asr model :gummy-chat-v1 is loaded
[asr-14] [INFO] [1755676559.954192471] [asr]: asr_node Initialization completed
[action_service_usb-13] [INFO] [1755676560.194134943] [action_service]: action service started...
[model_service-12] [INFO] [1755676561.034829238] [model_service]: LargeModelService node Initialization completed...
```

3.2 Test Case

This is a reference test case; users can create their own dialogue commands.

- Start tracking xx

Color/Face/Object/Machine Code/QR Code/Gesture Recognition/Human Posture

Color tracking, including: red, green, blue, and yellow (color calibration is required according to the **AI Large Model Preparation** tutorial).

Object Tracking

3.2.1 Example 1: "Start Tracking Red"

First, wake the robot with "Hi, yahboom" The robot responds: "I'm here, please." After the robot responds, the buzzer beeps briefly (beep—). The user can then speak. The robot will detect sound activity, printing 1 if there is sound activity and - if there is no sound activity. When the speech ends, it detects the end of the voice and stops recording if there is silence for more than 450ms.

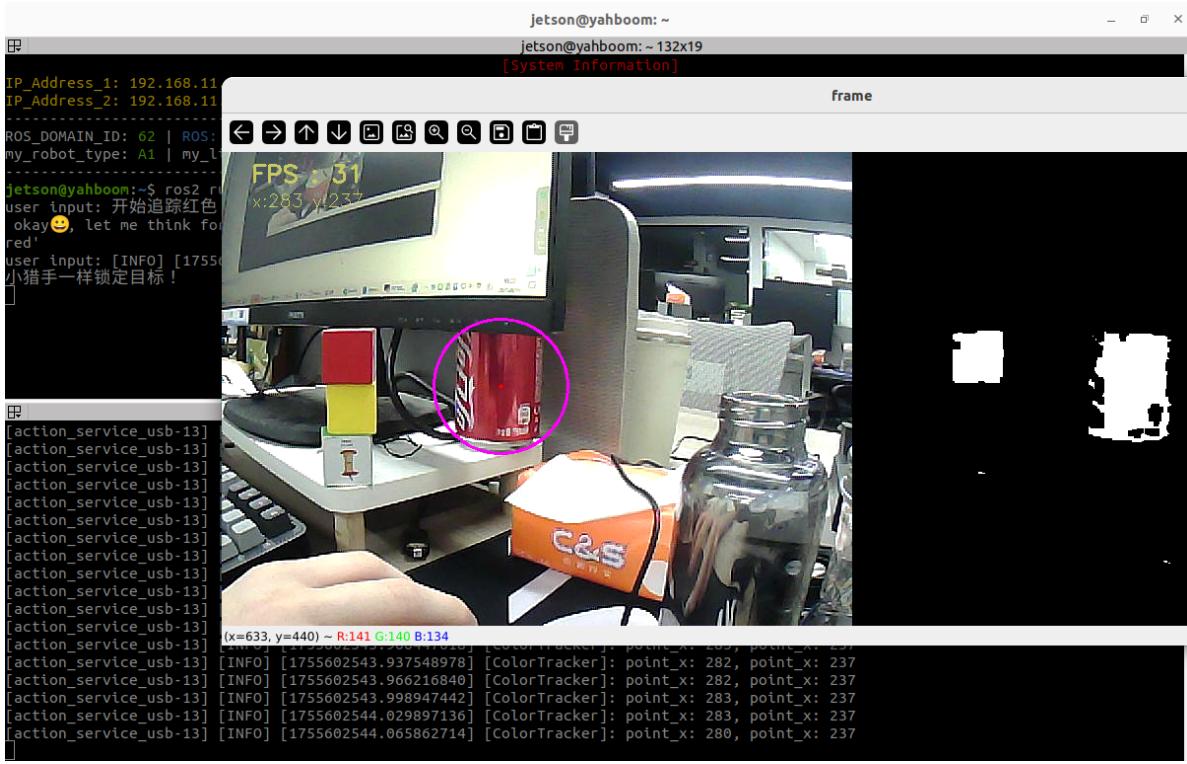
The following image shows Voice Active Detection (VAD):

```
jetson@yahboom: ~ 119x33
[asr-14] [INFO] [1755676560.194134943] [action_service]: action service started...
[model_service-12] [INFO] [1755676561.034829238] [model_service]: LargeModelService node Initialization completed...
[asr-14] [INFO] [1755677193.758468968] [asr]: I'm here
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
[asr-14] jack server is not running or cannot be started
[asr-14] JackShmReadWritePtr:::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] JackShmReadWritePtr:::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
[asr-14] jack server is not running or cannot be started
[asr-14] JackShmReadWritePtr:::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] JackShmReadWritePtr:::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
[asr-14] jack server is not running or cannot be started
[asr-14] JackShmReadWritePtr:::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] JackShmReadWritePtr:::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] [INFO] [1755677195.808943354] [asr]: 1
[asr-14] [INFO] [1755677195.869161395] [asr]: -
[asr-14] [INFO] [1755677195.930788959] [asr]: -
[asr-14] [INFO] [1755677195.990529768] [asr]: -
[asr-14] [INFO] [1755677196.051414359] [asr]: -
[asr-14] [INFO] [1755677196.110416196] [asr]: -
[asr-14] [INFO] [1755677196.169515509] [asr]: -
[asr-14] [INFO] [1755677196.230725039] [asr]: -
[asr-14] [INFO] [1755677196.291074443] [asr]: -
[asr-14] [INFO] [1755677196.351821782] [asr]: -
[asr-14] [INFO] [1755677196.421332818] [asr]: -
[asr-14] [INFO] [1755677196.482213345] [asr]: -
[asr-14] [INFO] [1755677196.542212305] [asr]: -
[asr-14] [INFO] [1755677196.603797368] [asr]: -
[asr-14] [INFO] [1755677196.664392989] [asr]: -
```

The robot will first communicate with the user, then respond to the user's instructions. The following information will be printed on the terminal:

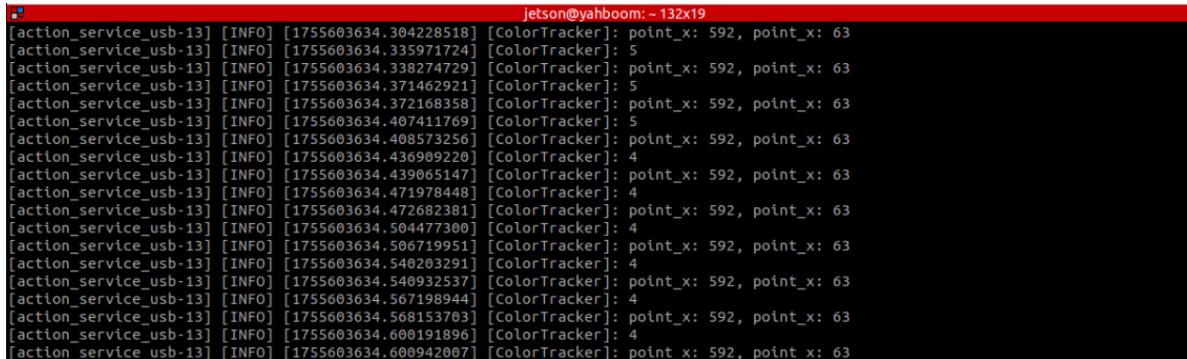
```
jetson@yahboom: ~ 122x34
[asr-14] [INFO] [1755689693.099078679] [asr]: -
[asr-14] [INFO] [1755689693.159681353] [asr]: -
[asr-14] [INFO] [1755689693.220133530] [asr]: -
[asr-14] [INFO] [1755689693.281055693] [asr]: -
[asr-14] [INFO] [1755689693.341707646] [asr]: -
[asr-14] [INFO] [1755689693.401669199] [asr]: -
[asr-14] [INFO] [1755689693.461263967] [asr]: -
[asr-14] [INFO] [1755689693.521016240] [asr]: -
[asr-14] [INFO] [1755689693.580646656] [asr]: -
[asr-14] [INFO] [1755689693.642947060] [asr]: -
[asr-14] [INFO] [1755689694.520350516] [asr]: 开始追踪红色。
[asr-14] [INFO] [1755689694.521760150] [asr]: 😊okay, let me think for a moment...
[model_service-12] [INFO] [1755689695.655800387] [model_service]: 决策层AI规划: The model service is abnormal. Check the large model account or configuration options
[model_service-12] [INFO] [1755689697.777113963] [model_service]: "action": ["colorTrack(red)"], "response": 好的，我开始追踪红色啦，就像一只专注的小猎手～
[action_service_usb-13] [INFO] [1755689704.504968251] [ColorTracker]: Loaded 4 color profiles
[action_service_usb-13] [INFO] [1755689704.509176996] [ColorTracker]: Red!
[action_service_usb-13] [INFO] [1755689704.556825478] [ColorTracker]: point_x: 15, point_x: 311
[action_service_usb-13] [INFO] [1755689704.832131391] [ColorTracker]: point_x: 15, point_x: 310
[action_service_usb-13] [INFO] [1755689704.84871617] [ColorTracker]: point_x: 17, point_x: 313
[action_service_usb-13] [INFO] [1755689704.874680759] [ColorTracker]: point_x: 17, point_x: 314
[action_service_usb-13] [INFO] [1755689704.910643073] [ColorTracker]: point_x: 15, point_x: 311
[action_service_usb-13] [INFO] [1755689704.944513991] [ColorTracker]: point_x: 17, point_x: 313
[action_service_usb-13] [INFO] [1755689704.973877124] [ColorTracker]: point_x: 16, point_x: 311
[action_service_usb-13] [INFO] [1755689705.007920650] [ColorTracker]: point_x: 16, point_x: 312
[action_service_usb-13] [INFO] [1755689705.044074583] [ColorTracker]: point_x: 16, point_x: 312
[action_service_usb-13] [INFO] [1755689705.073994103] [ColorTracker]: point_x: 16, point_x: 311
[action_service_usb-13] [INFO] [1755689705.108018656] [ColorTracker]: point_x: 16, point_x: 311
[action_service_usb-13] [INFO] [1755689705.138654017] [ColorTracker]: point_x: 16, point_x: 311
[action_service_usb-13] [INFO] [1755689705.179925945] [ColorTracker]: point_x: 16, point_x: 311
[action_service_usb-13] [INFO] [1755689705.205471887] [ColorTracker]: point_x: 16, point_x: 311
[action_service_usb-13] [INFO] [1755689705.241179067] [ColorTracker]: point_x: 16, point_x: 311
```

A window titled **frame** will open on the VNC screen, displaying the current robot's view.



Move the object slowly, and the servo and gimbal will follow.

If there is no target to track in the image, the program will count down for 10 seconds, a 5-second countdown will be printed on the terminal, and the process will automatically end, indicating the task is complete.



To manually end a task, wake the robot by saying "Hi, yahboom" The robot will respond, "I'm here, please." This will interrupt the program and automatically terminate the process, allowing you to proceed to the next command.

```

jetson@yahboom: ~ 122x34
[action_service_usb-13] [INFO] [1755690152.224030824] [ColorTracker]: point_x: 17, point_x: 203
[action_service_usb-13] [INFO] [1755690152.256612641] [ColorTracker]: point_x: 17, point_x: 203
[action_service_usb-13] [INFO] [1755690152.293598423] [ColorTracker]: point_x: 17, point_x: 203
[action_service_usb-13] [INFO] [1755690152.325380671] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.357234440] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.391917741] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.429926521] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.454495238] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.486866746] [ColorTracker]: point_x: 17, point_x: 202
[action_service_usb-13] [INFO] [1755690152.528392690] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.554341789] [ColorTracker]: point_x: 17, point_x: 203
[action_service_usb-13] [INFO] [1755690152.587647076] [ColorTracker]: point_x: 18, point_x: 204
[asr-14] [INFO] [1755690152.621888800] [asr]: I'm here
[action_service_usb-13] [INFO] [1755690152.625369419] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.638475139] [action_service]: 打断动作
[action_service_usb-13] [INFO] [1755690152.659189310] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] [INFO] [1755690152.664394573] [action_service]: killed process_pid
[action_service_usb-13] [INFO] [1755690152.693495739] [ColorTracker]: point_x: 18, point_x: 204
[action_service_usb-13] Failed to publish log message to rosout: publisher's context is invalid, at ./src/rcl/publisher.c:389
[action_service_usb-13] Exception in thread Thread-13565 (execute):
[action_service_usb-13] Traceback (most recent call last):
[action_service_usb-13]   File "/usr/lib/python3.10/threading.py", line 1016, in _bootstrap_inner
[action_service_usb-13]     self.run()
[action_service_usb-13]   File "/usr/lib/python3.10/threading.py", line 953, in run
[action_service_usb-13]     self._target(*self._args, **self._kwargs)
[action_service_usb-13]   File "/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_voice_ctrl/lib/python3.10/site-packages/yahboomcar_voice_ctrl/colorTracker.py", line 218, in execute
[action_service_usb-13]     self.pub_Servo.publish(self.servo_angle)
[action_service_usb-13]   File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/publisher.py", line 70, in publish
[action_service_usb-13]     self._publisher.publish(msg)
[action_service_usb-13] rclpy._rclpy_pybind11.RCLError: Failed to publish: publisher's context is invalid, at ./src/rcl/publisher.c:389

```

⌚ If the following warning appears when ending a trace, this is normal. It indicates that the child process has been terminated and will not affect normal program operation.

```

[action_service_usb-13] Failed to publish log message to rosout: publisher's
context is invalid, at ./src/rcl/publisher.c:389
[action_service_usb-13] Exception in thread Thread-284 (execute):
[action_service_usb-13] rclpy._rclpy_pybind11.RCLError: Failed to publish:
publisher's context is invalid, at ./src/rcl/publisher.c:389

```

The robot now enters the free conversation state again, but all conversation history is retained. You can wake yahboom up again and use "End Current Task" to terminate the current task cycle, clear the conversation history, and start a new one.

```

jetson@yahboom: ~ 122x34
[asr-14] [INFO] [1755679669.327887218] [asr]: 1
[asr-14] [INFO] [1755679669.388356288] [asr]: 1
[asr-14] [INFO] [1755679669.448540247] [asr]: 1
[asr-14] [INFO] [1755679669.509674586] [asr]: 1
[asr-14] [INFO] [1755679669.571171770] [asr]: 1
[asr-14] [INFO] [1755679669.630793924] [asr]: -
[asr-14] [INFO] [1755679669.691181227] [asr]: -
[asr-14] [INFO] [1755679669.751568947] [asr]: -
[asr-14] [INFO] [1755679669.812568907] [asr]: -
[asr-14] [INFO] [1755679669.873032793] [asr]: -
[asr-14] [INFO] [1755679669.933673172] [asr]: -
[asr-14] [INFO] [1755679669.994094078] [asr]: -
[asr-14] [INFO] [1755679670.024206570] [asr]: -
[asr-14] [INFO] [1755679670.084441603] [asr]: -
[asr-14] [INFO] [1755679670.145278316] [asr]: -
[asr-14] [INFO] [1755679670.205037150] [asr]: -
[asr-14] [INFO] [1755679670.266176063] [asr]: -
[asr-14] [INFO] [1755679670.326090750] [asr]: -
[asr-14] [INFO] [1755679670.386630159] [asr]: -
[asr-14] [INFO] [1755679670.446504875] [asr]: -
[asr-14] [INFO] [1755679670.506955765] [asr]: -
[asr-14] [INFO] [1755679670.567423776] [asr]: -
[asr-14] [INFO] [1755679670.628749552] [asr]: -
[asr-14] [INFO] [1755679670.688834205] [asr]: -
[asr-14] [INFO] [1755679670.750200240] [asr]: -
[asr-14] [INFO] [1755679670.809695021] [asr]: -
[asr-14] [INFO] [1755679670.870388459] [asr]: -
[asr-14] [INFO] [1755679671.689391598] [asr]: 结束当前任务。
[asr-14] [INFO] [1755679671.691073141] [asr]: 😊okay, let me think for a moment...
[model_service-12] [INFO] [1755679673.748230368] [model_service]: "action": ["finish_dialogue()"], "response": 好的，任务
已经结束了，我这就退下休息啦，有需要再叫我哦～
[model_service-12] [INFO] [1755679679.725228752] [model_service]: The current instruction cycle has ended
[action_service_usb-13] [INFO] [1755679679.725376348] [action_service]: Published message: finish

```

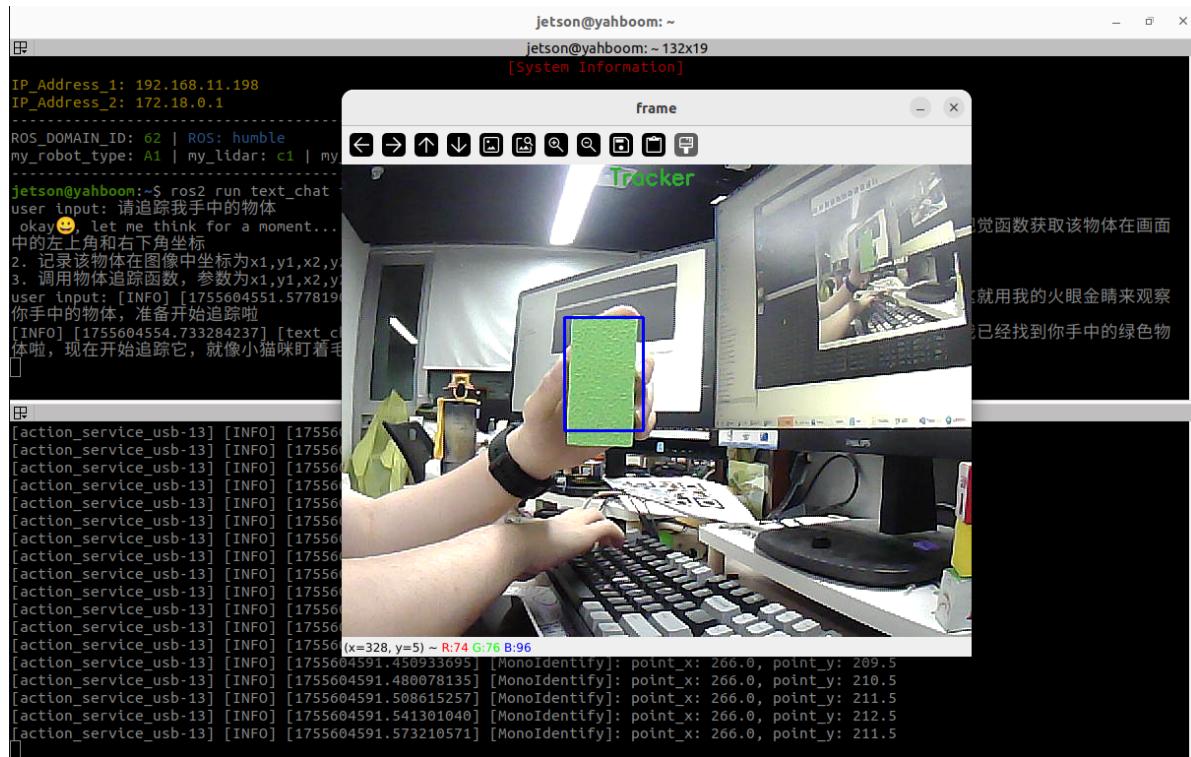
3.2.2 Example 2: "Please track the object in my hand"

⚠ The coordinates obtained in this example are entirely derived from the inference of a large AI model. Therefore, it is recommended to use a newer model for better results!

Similar to the test in Example 1, first wake the robot with "Hi, yahboom" After the robot responds, the buzzer will briefly beep. The user can then speak. After speaking, the robot will respond and move according to the user's instructions. Hold any object in its field of view and hold until the tracking box appears.

```
jetson@yahboom: ~ 132x39
[asr-14] [INFO] [1755691213.302513700] [asr]: -
[asr-14] [INFO] [1755691213.362415638] [asr]: -
[asr-14] [INFO] [1755691213.422893915] [asr]: -
[asr-14] [INFO] [1755691213.484933397] [asr]: -
[asr-14] [INFO] [1755691213.543828357] [asr]: -
[asr-14] [INFO] [1755691213.604607733] [asr]: -
[asr-14] [INFO] [1755691213.663974965] [asr]: -
[asr-14] [INFO] [1755691213.724576959] [asr]: -
[asr-14] [INFO] [1755691213.784176935] [asr]: -
[asr-14] [INFO] [1755691213.844814162] [asr]: -
[asr-14] [INFO] [1755691213.905347705] [asr]: -
[asr-14] [INFO] [1755691213.966867521] [asr]: -
[asr-14] [INFO] [1755691214.026808853] [asr]: -
[asr-14] [INFO] [1755691214.086665094] [asr]: -
[asr-14] [INFO] [1755691214.147043336] [asr]: -
[asr-14] [INFO] [1755691215.296358428] [asr]: 请追踪我手中的物体。
[asr-14] [INFO] [1755691215.297772300] [asr]: 😊 okay, let me think for a moment...
[model_service-12] [INFO] [1755691217.641956821] [model_service]: "action": ['seewhat()'], "response": 好的，我准备开始追踪你手中的物体啦，先让我看看清楚-
[model_service-12] [INFO] [1755691225.962772389] [model_service]: "action": ['monoTracker(150, 40, 230, 140)'], "response": 我看到你手中的绿色海绵了，现在开始追踪它啦！
[action_service_usb-13] [INFO] [1755691233.464004128] [MonoIdentify]: Initializing tracker with ROI: (150, 40, 80, 100)
[action_service_usb-13] [INFO] [1755691233.470355765] [MonoIdentify]: point_x: 190.0, point_y: 90.0
[action_service_usb-13] [INFO] [1755691233.739016795] [MonoIdentify]: point_x: 190.0, point_y: 90.0
[action_service_usb-13] [INFO] [1755691233.762328455] [MonoIdentify]: point_x: 190.0, point_y: 90.0
[action_service_usb-13] [INFO] [1755691233.790096585] [MonoIdentify]: point_x: 190.0, point_y: 90.0
[action_service_usb-13] [INFO] [1755691233.827770871] [MonoIdentify]: point_x: 190.0, point_y: 90.0
```

A window titled **frame** will open on the VNC screen, displaying the current robot's view.



Move the object slowly, and the servo and gimbal will follow.

If there is no target to track in the image, the program will count down for 10 seconds, a 5-second countdown will be printed on the terminal, and the process will automatically end, indicating the task is complete. Success

```
jetson@yahboom: ~ 132x19
[action_service_usb-13] [INFO] [1755605055.316859319] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.346928043] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.375728720] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.406246064] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.435578061] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.467338815] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.498990989] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.526724746] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.557201133] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.586995666] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.618767081] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.647714451] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.678264505] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.705587015] [MonoIdentify]: 1
[action_service_usb-13] [INFO] [1755605055.744913680] [action_service]: Published message: 机器人反馈:执行追踪任务完成
[action_service_usb-13] [INFO] [1755605055.779898713] [action_service]: killed process_pid
[action_service_usb-13] Waiting for at least 1 matching subscription(s)...
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: yahboomcar_msgs.msg.ServoControl(s1=90, s2=35)
```

To manually end a task, wake up the robot with the voice command "Hi, yahboom" The robot will respond, "I'm here, please." This will interrupt the program and automatically terminate the process, allowing you to proceed to the next command.

```
jetson@yahboom: ~ 132x39
[action_service_usb-13] [INFO] [1755691622.543743568] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.574027073] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.603133322] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.638109181] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.666522957] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.694625043] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.727612097] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.754693924] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.783917617] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.815799705] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[asr-14] [INFO] [1755691622.825632363] [asr]: I'm here
[action_service_usb-13] [INFO] [1755691622.828855162] [action_service]: 打断动作
[action_service_usb-13] [INFO] [1755691622.844219338] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.855146498] [action_service]: killed process_pid
[action_service_usb-13] [INFO] [1755691622.878110391] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] [INFO] [1755691622.903534369] [MonoIdentify]: point_x: 308.0, point_y: 234.0
[action_service_usb-13] Failed to publish log message to rosout: publisher's context is invalid, at ./src/rcl/publisher.c:389
[action_service_usb-13] Exception in thread Thread-287 (execute):
[action_service_usb-13] Traceback (most recent call last):
[action_service_usb-13]   File "/usr/lib/python3.10/threading.py", line 1016, in _bootstrap_inner
[action_service_usb-13]     self.run()
[action_service_usb-13]   File "/usr/lib/python3.10/threading.py", line 953, in run
[action_service_usb-13]     self._target(*self._args, **self._kwargs)
[action_service_usb-13]   File "/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_voice_ctrl/lib/python3.10/site-packages/yahboomcar_voice_ctrl/monoTracker.py", line 184, in execute
[action_service_usb-13]     self.pub_Servo.publish(self.servo_angle)
[action_service_usb-13]   File "/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/publisher.py", line 70, in publish
[action_service_usb-13]     self._publisher.publish(msg)
[action_service_usb-13] rclpy.rclpy_pybind11.RCLError: Failed to publish: publisher's context is invalid, at ./src/rcl/publisher.c:389
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: yahboomcar_msgs.msg.ServoControl(s1=90, s2=35)
[action_service_usb-13]
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
[asr-14] jack server is not running or cannot be started
[asr-14] JackShmReadWritePtr::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] JackShmReadWritePtr::JackShmReadWritePtr - Init not done for -1, skipping unlock
[asr-14] Cannot connect to server socket err = No such file or directory
[asr-14] Cannot connect to server request channel
```

The robot will now enter a free conversation mode, but all conversation history will be retained. You can wake yahboom up again and select "End Current Task" to have the robot end the current task cycle, clear the conversation history, and start a new one.

```
jetson@yahboom: ~ 122x34
[asr-14] [INFO] [1755679669.327887218] [asr]: 1
[asr-14] [INFO] [1755679669.388356288] [asr]: 1
[asr-14] [INFO] [1755679669.448540247] [asr]: 1
[asr-14] [INFO] [1755679669.509674586] [asr]: 1
[asr-14] [INFO] [1755679669.571171770] [asr]: 1
[asr-14] [INFO] [1755679669.630793924] [asr]: -
[asr-14] [INFO] [1755679669.691181227] [asr]: -
[asr-14] [INFO] [1755679669.751568947] [asr]: -
[asr-14] [INFO] [1755679669.812568907] [asr]: -
[asr-14] [INFO] [1755679669.873032793] [asr]: -
[asr-14] [INFO] [1755679669.933673172] [asr]: -
[asr-14] [INFO] [1755679669.994094078] [asr]: -
[asr-14] [INFO] [1755679670.024206570] [asr]: -
[asr-14] [INFO] [1755679670.084441603] [asr]: -
[asr-14] [INFO] [1755679670.145278316] [asr]: -
[asr-14] [INFO] [1755679670.205037150] [asr]: -
[asr-14] [INFO] [1755679670.266176063] [asr]: -
[asr-14] [INFO] [1755679670.326090750] [asr]: -
[asr-14] [INFO] [1755679670.386630159] [asr]: -
[asr-14] [INFO] [1755679670.446504875] [asr]: -
[asr-14] [INFO] [1755679670.506955765] [asr]: -
[asr-14] [INFO] [1755679670.567423776] [asr]: -
[asr-14] [INFO] [1755679670.628749552] [asr]: -
[asr-14] [INFO] [1755679670.688834205] [asr]: -
[asr-14] [INFO] [1755679670.750200240] [asr]: -
[asr-14] [INFO] [1755679670.809695021] [asr]: -
[asr-14] [INFO] [1755679670.870388459] [asr]: -
[asr-14] [INFO] [1755679671.689391598] [asr]: 结束当前任务。
[asr-14] [INFO] [1755679671.691073141] [asr]: 😊okay, let me think for a moment...
[model_service-12] [INFO] [1755679673.748230368] [model_service]: "action": [finish_dialogue()], "response": 好的，任务已经结束了，我这就退下休息啦，有需要再叫我哦～
[model_service-12] [INFO] [1755679679.725228752] [model_service]: The current instruction cycle has ended
[action_service_usb-13] [INFO] [1755679679.725376348] [action_service]: Published message: finish
```

4. Source Code Analysis

Source code location:

Jetson Orin Nano:

```
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemode1/largemode1/action_s  
ervice_usb.py
```

RDK X5:

```
/home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/largemode1/largemode1/action_  
service_usb.py
```

Jetson Nano, Raspberry Pi host:

You need to enter Docker first.

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/largemode1/largemode1/action_service_  
usb.py
```

4.1 Example 1

action_service_usb.py :

Example 1 uses the **seewhat**, **color_follow**, and **stop_track** methods in the **CustomActionServer** class.

- The **seewhat** function primarily retrieves the camera's color image.
- The **colorTrack(self, color)** function performs color tracking.
- The **stop_track()** function issues a stop tracking command.

This section focuses on the **colorTrack(self, color)** function, which requires a color parameter, which can be 'red', 'green', 'blue', or 'yellow'.

```
[asr-14] [INFO] [1755689694.520350516] [asr]: 开始追踪红色。  
[asr-14] [INFO] [1755689694.521760150] [asr]: 😊Okay, let me think for a moment...  
[model_service-12] [INFO] [1755689695.655800387] [model_service]: 决策层AI规划: The model service is abnormal. Check the la  
ge model account or configuration options  
[model_service-12] [INFO] [1755689697.777113963] [model_service]: "action": ["colorTrack(red)"], "response": 好的，我开始  
追踪红色啦，就像一只专注的小猎手~
```

```
# Start the color tracking subprocess
process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl',
    'colorTracker', '--ros-args', '-p', f'target_color:={target_color}'])
```

The startup program source code is located at:

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voice_ctrl/  
colorTracker.py
```

```
def colorTrack(self,color):
    try:
        self.colorTracker_future = Future()
        color = color.strip("\\"")
        if color == 'red':
            target_color = int(1)
        elif color == 'green':
```

```

        target_color = int(2)
    elif color == 'blue':
        target_color = int(3)
    elif color == 'yellow':
        target_color = int(4)
    else:
        target_color = int(1)
    process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl',
'colorTracker', '--ros-args', '-p', f'target_color:={target_color}'])
    while not self.colorTracker_future.done():
        if self.interrupt_flag:
            break
        time.sleep(0.1)
    self.get_logger().info(f'killed process_{process_1.pid}')
    self.kill_process_tree(process_1.pid)
    self.cancel()
except:
    self.get_logger().error('colorTrack Startup failure')
return

```

When the main model receives a user input command for "Stop Tracking" or "End Tracking," or if the tracking target is lost for more than 10 seconds,

the **stop_track** method is called, sending the future.done signal. The `while not self.colorTracker_future.done()` block in the **colorTrack** function then exits. The **kill_process_tree** method is then called to recursively kill the child process tree. Finally, the status of the action execution is reported back to the main model at the execution layer.

4.2 Example 2

action_service_usb.py:

Example 2 uses the **seewhat**, **monoTracker**, and **stop_track** methods in the **CustomActionServer** class.

- The **seewhat** function primarily retrieves the camera's color image.
- The **monoTracker(self,x1,y1,x2,y2)** function performs object tracking. - **stop_track()** issues a tracking stop command.

The **seewhat** function primarily retrieves the camera's color image. The **monoTracker(self,x1,y1,x2,y2)** function takes as parameters the coordinates of the upper-left and lower-right vertices of the object's bounding box (the upper-left corner of the image is the pixel origin). For example, the coordinates of the outer bounding box of the identified green square in Example 2 can be found from the large model's response: the upper-left corner is (240,145) and the lower-right corner is (320,260).

```

[asr-14] [INFO] [1755691215.296358428] [asr]: 请追踪我手中的物体。
[asr-14] [INFO] [1755691215.297772300] [asr]: 😊okay, let me think for a moment...
[model_service-12] [INFO] [1755691217.641956821] [model_service]: "action": ["seewhat()"], "response": 好的，我准备开始追踪你手中的物体啦，先让我看看清楚-
[model_service-12] [INFO] [1755691225.962772389] [model_service]: "action": ["monoTracker(150, 40, 230, 140)"], "response": 我看到你手中的绿色海绵了，现在开始追踪它啦！
[action_service_usb-13] [INFO] [1755691233.464004128] [MonoIdentify]: Initializing tracker with ROI: (150, 40, 80, 100)

```

```

#Start the object tracking subprocess program
process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl',
'monoTracker', '--ros-args', '-p', f'x1:={x1}', '-p', f'y1:={y1}', '-p', f'x2:={x2}', '-p', f'y2:={y2}'])

```

The startup program source code path is:

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voice_ctrl/
monoTracker.py
```

```
def monoTracker(self,x1,y1,x2,y2):
    try:
        self.monoTracker_future = Future()
        x1 = int(x1)
        x2 = int(x2)
        y1 = int(y1)
        y2 = int(y2)
        process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl',
        'monoTracker', '--ros-args', '-p', f'x1:={x1}', '-p', f'y1:={y1}', '-p', f'x2:={x2}', '-p',
        f'y2:={y2}'])
        while not self.monoTracker_future.done():
            if self.interrupt_flag:
                break
            time.sleep(0.1)
        self.get_logger().info(f'killed process_pid')
        self.kill_process_tree(process_1.pid)
        self.cancel()
    except:
        self.get_logger().error('monoTracker startup failure')
    return
```

When the main model receives a user input command for "Stop Tracking" or "End Tracking," or if the tracking target is lost for more than 10 seconds,

The **stop_track** method will be called to send the `future.done` signal. After that, `while not self.monoTracker_future.done()` in the **monoTracker** function will exit the blocking state. Then the **kill_process_tree** method will be called to recursively kill the process tree of the child process. Finally, the status of the execution action will be fed back to the execution layer model.