

# Multimodal visual understand + visual line patrolling(Voice Version)

---

## Multimodal visual understand + visual line patrolling(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: "Start Green Line Follow"
4. Source Code Analysis
  - 4.1 USB Camera, action\_service\_usb.py
  - 4.2 nuwa Depth Camera, action\_service\_nuwa.py

## 1. Course Content

---


1. Learn to use the robot's visual understanding capabilities in conjunction with line-following autonomous driving.
2. Analyze newly discovered 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 PI5 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 largemodel largemodel_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 largemodel largemodel_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 @ 0xaaab21f59650] 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] YUVV 4:2:2 1280 x 720 (9 Hz)
[usb_cam_node_exe-1] YUVV 4:2:2 1920 x 1080 (6 Hz)
[usb_cam_node_exe-1] YUVV 4:2:2 1024 x 768 (6 Hz)
[usb_cam_node_exe-1] YUVV 4:2:2 640 x 480 (30 Hz)
[usb_cam_node_exe-1] YUVV 4:2:2 800 x 600 (20 Hz)
[usb_cam_node_exe-1] YUVV 4:2:2 1280 x 1024 (6 Hz)
[usb_cam_node_exe-1] YUVV 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 color line follow
    - Colors include: red, green, blue, and yellow (color calibration is required according to the **AI Large Model Preparation** tutorial).

### 3.2.1 Example: "Start Green Line Follow"

First, use "Hi, yahboom" to wake the robot. The robot responds: "I'm here, please tell me." After the robot responds, the buzzer beeps briefly (beep—). The user can speak, and the robot will detect sound activity. If there is sound activity, it will print 1; if there is no sound activity, it will print -. When the speech ends, it will detect the end of the voice and stop recording if there is

silence for more than 450ms.

The dynamic sound detection (VAD) is shown in the image below:

```
jetson@yahboom: ~ 119x33
[asr-14] [INFO] [1755677193.758468968] [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...
[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]: 1
[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]: -
```

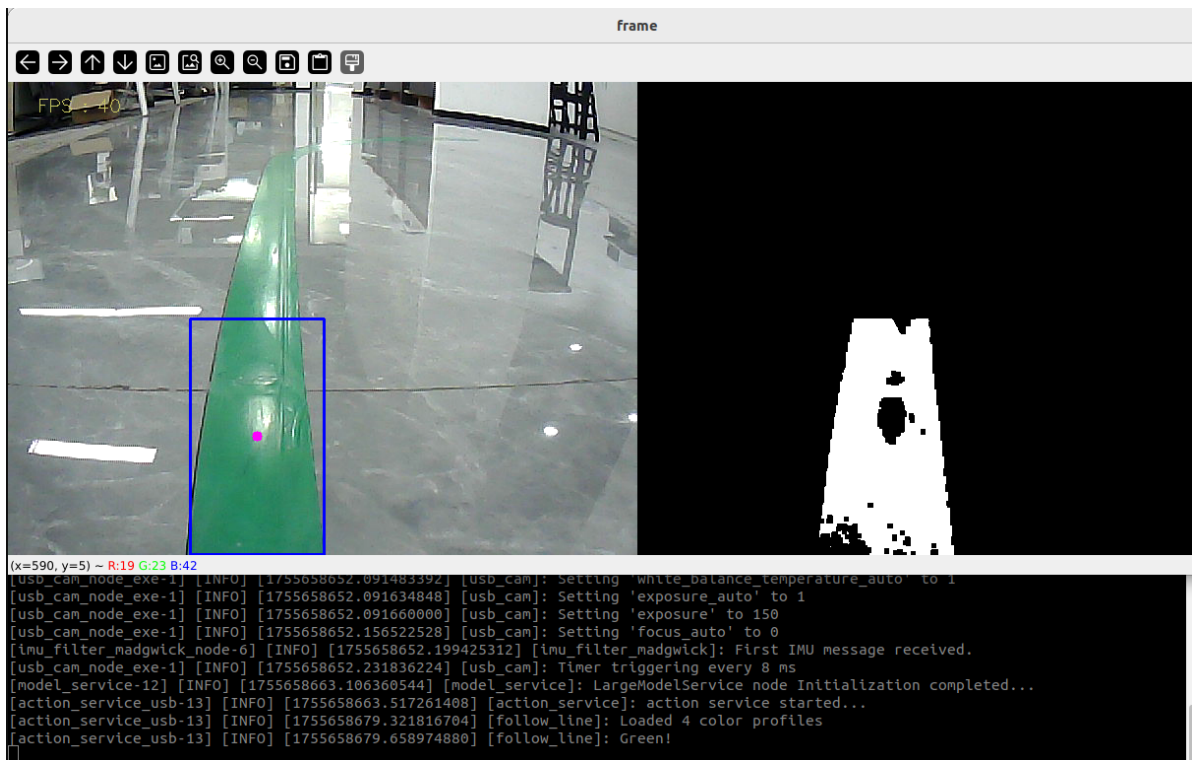
The robot will first communicate with the user, then respond to the user's instructions. The terminal will also print the following information:

For users of the gimbal servo USB camera version, upon program startup, the servo gimbal will default to pointing downward.

Depth Camera users need to adjust the camera angle appropriately.

```
jetson@yahboom: ~ 132x39
[asr-14] [INFO] [1755747813.536605297] [asr]: -
[asr-14] [INFO] [1755747813.596123002] [asr]: -
[asr-14] [INFO] [1755747813.655121040] [asr]: -
[asr-14] [INFO] [1755747813.716772100] [asr]: -
[asr-14] [INFO] [1755747813.775781755] [asr]: -
[asr-14] [INFO] [1755747813.836481262] [asr]: -
[asr-14] [INFO] [1755747813.897148703] [asr]: -
[asr-14] [INFO] [1755747813.956894736] [asr]: -
[asr-14] [INFO] [1755747814.017704134] [asr]: -
[asr-14] [INFO] [1755747814.076621719] [asr]: -
[asr-14] [INFO] [1755747814.107143205] [asr]: -
[asr-14] [INFO] [1755747814.167740625] [asr]: -
[asr-14] [INFO] [1755747814.233259785] [asr]: -
[asr-14] [INFO] [1755747814.293910487] [asr]: -
[asr-14] [INFO] [1755747814.353881933] [asr]: -
[asr-14] [INFO] [1755747814.415238548] [asr]: -
[asr-14] [INFO] [1755747814.475879746] [asr]: -
[asr-14] [INFO] [1755747814.536313416] [asr]: -
[asr-14] [INFO] [1755747814.595454465] [asr]: -
[asr-14] [INFO] [1755747815.520222280] [asr]: 开始寻绿线。
[asr-14] [INFO] [1755747815.520930688] [asr]: 😊okay, let me think for a moment...
[model_service-12] [INFO] [1755747816.639811554] [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] [1755747822.304740769] [model_service]: "action": ['follow_line(green)'], "response": 好的, 我马上开始寻绿线自动驾驶, 就像一个专业的赛车手一样!
[action_service_usb-13] [INFO] [1755747830.567033514] [follow_line]: Loaded 4 color profiles
[action_service_usb-13] [INFO] [1755747830.573573962] [follow_line]: Green!
```

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



Then, speed calculation begins, and the robot automatically patrols the line. If it encounters an obstacle during patrol, it stops and the buzzer sounds.

If there are no targets to track on screen, 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 mission is complete.

```

[action_service_usb-13] [INFO] [1755747840.356931445] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.381208236] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.406432387] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.433669663] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.459813750] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.480075397] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.504316123] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.534303763] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.554777834] [follow_line]: 1
[action_service_usb-13] [INFO] [1755747840.585265012] [action_service]: Published message: 机器人反馈:执行追踪任务完成
[action_service_usb-13] [INFO] [1755747840.605077235] [action_service]: killed process_pid
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
[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)
[action_service_usb-13]
[model_service-12] [INFO] [1755747844.437551800] [model service]: "action": ['finishtask()'], "response": 我已经完成了寻绿线的任务, 感觉就像完成了一场精彩的赛车比赛! 还有什么我可以帮忙的吗?
[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: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
[action_service_usb-13] Waiting for at least 1 matching subscription(s)...
[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)
[action_service_usb-13]

```

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

```
[asr-14] [INFO] [1755748099.689721364] [asr]: -
[asr-14] [INFO] [1755748100.541226622] [asr]: 开始寻绿线。
[asr-14] [INFO] [1755748100.541914609] [asr]: 😊Okay, let me think for a moment...
[model_service-12] [INFO] [1755748103.044057416] [model_service]: "action": ['follow_line(green)'], "response": 好的，我马上开始寻绿线自动驾驶，就像一个专业的赛车手一样！
[action_service_usb-13] [INFO] [1755748111.217803257] [follow_line]: Loaded 4 color profiles
[action_service_usb-13] [INFO] [1755748111.223258220] [follow_line]: Green!
[asr-14] [INFO] [1755748115.027715952] [asr]: I'm here
[action_service_usb-13] [INFO] [1755748115.029802632] [action_service]: 打断动作
[action_service_usb-13] [INFO] [1755748115.094859197] [action_service]: killed process_pid
[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] [1755748116.936547765] [asr]: 1
[action_service_usb-13] publisher: beginning loop
[action_service_usb-13] publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
[action_service_usb-13]
[asr-14] [INFO] [1755748116.907308346] [asr]: 1
```

The robot will now enter a free conversation state, but all conversation history will be retained. You can wake Xiaoya up again and select "End Current Mission" to end the current mission 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 host:

```
#NUWA camera user
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_nuwa.py
#USB camera user
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_usb.py
```

RDk X5:



```
#NUWA camera user
/home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_
service_nuwa.py
#USB camera user
/home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_
service_usb.py
```

Jetson Nano, Raspberry Pi host:

You need to first enter Docker.

```
#NUWA Camera User
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_
nuwa.py
#USB Camera User
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_service_
usb.py
```

## 4.1 USB Camera, action\_service\_usb.py

- Example 1 uses the **seewhat**, **follow\_line(self, color)**, and **stop\_track** methods in the **CustomActionServer** class.
  - The **seewhat** function primarily retrieves the camera's color image.
  - The **follow\_line(self, color)** function performs color line tracking.
  - **stop\_track()** issues a stop tracking command function.

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

```
[asr-14] [INFO] [1755748100.541226622] [asr]: 开始寻绿线。
[asr-14] [INFO] [1755748100.541914609] [asr]: 😊okay, let me think for a moment...
[model_service-12] [INFO] [1755748103.044057416] [model_service]: "action": ['follow_line(green)'], "response": 好的，我马上开始寻绿
线自动驾驶，就像一个专业的赛车手一样！
[action_service_usb-13] [INFO] [1755748111.217803257] [follow_line]: Loaded 4 color profiles
```

```
# Start the line-following autonomous driving subprocess program
process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl',
'follow_line', '--ros-args', '-p', f'target_color:={target_color}'])
```

The startup program source code path is:

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

```
def follow_line(self,color):
    try:
        self.follow_line_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',
        'follow_line', '--ros-args', '-p', f'target_color:={target_color}'])
        while not self.follow_line_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('follow_line Startup failure')
    return

```

When the main model receives the user input of the [Stop Tracking] or [End Tracking] command, 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.follow_line_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 to the main model at the execution layer.

## 4.2 nuwa Depth Camera, action\_service\_nuwa.py

- Example 1 uses the **seewhat**, **follow\_line(self, color)**, and **stop\_follow** methods in the **CustomActionServer** class.
  - The **seewhat** function primarily retrieves the color image from the depth camera.
  - The **follow\_line(self, color)** function performs color line tracking.
  - **stop\_follow()** issues a stop command to follow the function.

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

```

# Start the line-following autonomous driving subprocess program.
process_1= subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl_depth',
'voice_follow_line', '--ros-args', '-p', f'color:={target_color}'])

```

```

[asr-14] [INFO] [1755748100.541226622] [asr]: 开始寻绿线。
[asr-14] [INFO] [1755748100.541914609] [asr]: 😊okay, let me think for a moment...
[model_service-12] [INFO] [1755748103.044057416] [model_service]: "action": ["follow_line(green)"], "response": "好的，我马上开始寻绿线自动驾驶，就像一个专业的赛车手一样！"
[action_service_usb-13] [INFO] [1755748111.217803257] [follow_line]: Loaded 4 color profiles

```

The startup program source code path is:

```

~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl_depth/yahboomcar_voice_ctrl_depth/follow/voice_follow_line.py

```

```

def follow_line(self,color):
    self.follow_line_future = Future() #Reset Future object
    color = color.strip('"') # Remove single and double quotes
    if color == 'red':
        target_color = float(1)
    elif color == 'green':
        target_color = float(2)
    elif color == 'blue':
        target_color = float(3)
    elif color == 'yellow':
        target_color = float(4)

```

```

else:
    self.get_logger().info('color_sort:error')
    return
    process_1= subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl_depth',
'voice_follow_line', '--ros-args', '-p', f'colcor:={target_color}'])
    while not self.follow_line_future.done():
        if self.interrupt_flag:
            break
        time.sleep(0.1)

    self.kill_process_tree(process_1.pid)
    self.cancel()

```

When the main model receives the user input of the "Stop Following" or "End Following" command,

or if the tracking target is lost for more than 10 seconds,

the **stop\_follow** method is called, sending the future.done signal. The `while not self.follow_line_future.done()` block in the **follow\_line** function is then exited. The **kill\_process\_tree** method is then called to recursively kill the child process tree. Finally, the status of the execution action is reported to the main model at the execution layer.