Multimodal visual understand + visual following (Text Version)

Multimodal visual understand + visual following (Text Version)
1. Course Content
2. Preparation
2.1 Content Description
3. Running the Example
3.1 Starting the Program
3.2 Test Cases
3.2.1 Example 1: "Start following red"
3.2.2 Example 2: "Follow the Object in My Hand" (Depth Camera)
4. Source Code Parsing
4.1 Example 1
4.1.1 Nuwa Depth Camera, action_service_nuwa.py
4.1.2 USB Camera, action_service_usb.py
4.2 Example 2
4.2.1 nuwa Depth Camera, action_service_nuwa.py

1. Course Content

1. Learn to use the robot's vision combined with the robot's follow-up function
2. Analyze new key source code

2. Preparation

2.1 Content Description

> 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.

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

⚠ The responses from the large model may not be exactly the same for the same test command and may differ slightly from the screenshots in the tutorial. To increase or decrease the diversity of the large model's responses, refer to the section on configuring the decision-making large 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 example. The functionality is largely the same, so I will not elaborate on the debugging details of the program. The results will be detailed here! ! !

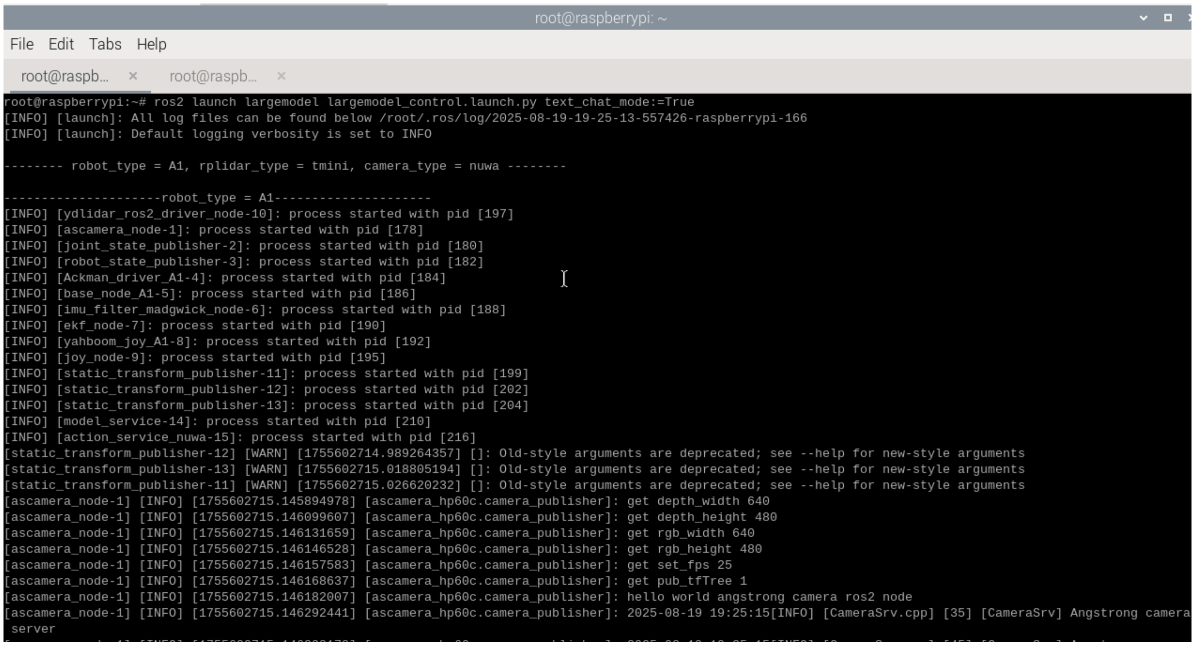
3. Running the Example

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 in Docker and enter the following command:

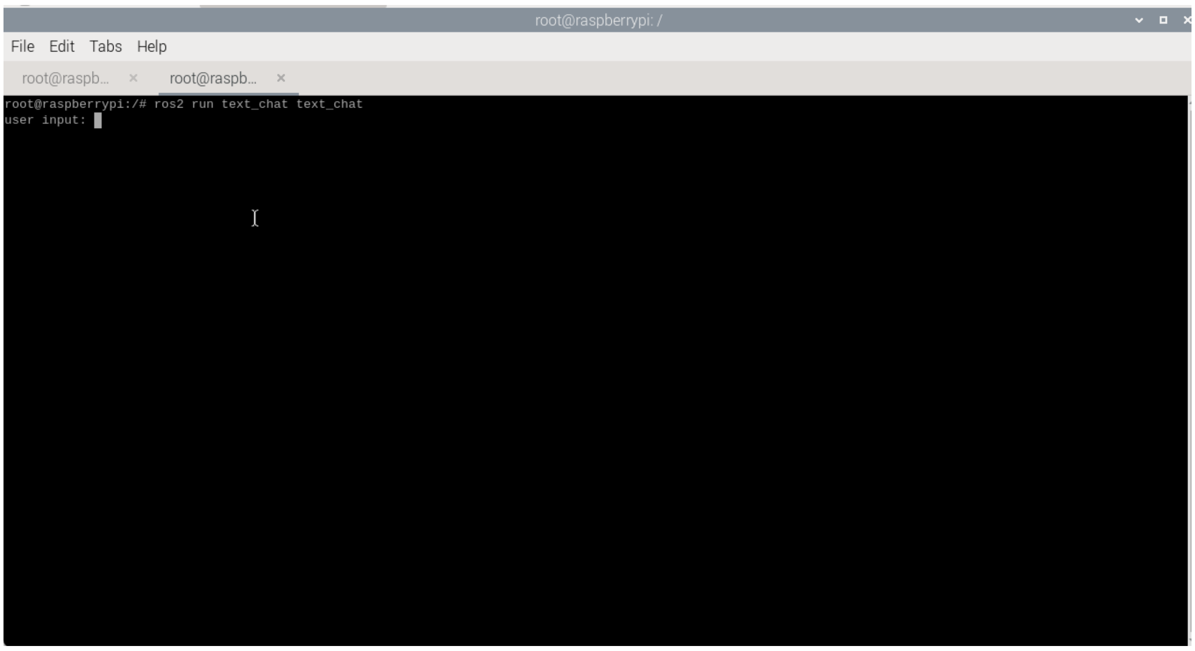
```
ros2 launch largemodel largemodel_control.launch.py text_chat_mode:=True
```



```
root@raspberrypi:~# ros2 launch largemodel largemodel_control.launch.py text_chat_mode:=True
[INFO] [launch]: All log files can be found below /root/.ros/log/2025-08-19-19-25-13-557426-raspberrypi-166
[INFO] [launch]: Default logging verbosity is set to INFO

-------- robot_type = A1, rplidar_type = tmini, camera_type = nuwa --------

---------------------robot_type = A1---------------------
[INFO] [ydlidar_ros2_driver_node-10]: process started with pid [197]
[INFO] [ascamera_node-1]: process started with pid [178]
[INFO] [joint_state_publisher-2]: process started with pid [180]
[INFO] [robot_state_publisher-3]: process started with pid [182]
[INFO] [Ackman_driver_A1-4]: process started with pid [184]
[INFO] [base_node_A1-5]: process started with pid [186]
[INFO] [imu_filter_madgwick_node-6]: process started with pid [188]
[INFO] [ekf_node-7]: process started with pid [190]
[INFO] [yahboom_joy_A1-8]: process started with pid [192]
[INFO] [joy_node-9]: process started with pid [195]
[INFO] [static_transform_publisher-11]: process started with pid [199]
[INFO] [static_transform_publisher-12]: process started with pid [202]
[INFO] [static_transform_publisher-13]: process started with pid [204]
[INFO] [model_service-14]: process started with pid [210]
[INFO] [action_service_nuwa-15]: process started with pid [216]
[static_transform_publisher-12] [WARN] [1755602714.989264357] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-13] [WARN] [1755602715.018805194] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-11] [WARN] [1755602715.026620232] []: Old-style arguments are deprecated; see --help for new-style arguments
[ascamera_node-1] [INFO] [1755602715.145894978] [ascamera_hp60c.camera_publisher]: get depth_width 640
[ascamera_node-1] [INFO] [1755602715.146099607] [ascamera_hp60c.camera_publisher]: get depth_height 480
[ascamera_node-1] [INFO] [1755602715.146131659] [ascamera_hp60c.camera_publisher]: get rgb_width 640
[ascamera_node-1] [INFO] [1755602715.146146528] [ascamera_hp60c.camera_publisher]: get rgb_height 480
[ascamera_node-1] [INFO] [1755602715.146157583] [ascamera_hp60c.camera_publisher]: get set_fps 25
[ascamera_node-1] [INFO] [1755602715.146168637] [ascamera_hp60c.camera_publisher]: get pub_tfTree 1
[ascamera_node-1] [INFO] [1755602715.146182007] [ascamera_hp60c.camera_publisher]: hello world angstrong camera ros2 node
[ascamera_node-1] [INFO] [1755602715.146292441] [ascamera_hp60c.camera_publisher]: 2025-08-19 19:25:15[INFO] [CameraSrv.cpp] [35] [CameraSrv] Angstrong camera
 server
```

Open the same Docker container in multiple terminals and start it.

```
ros2 run text_chat text_chat
```



```
root@raspberrypi:/# ros2 run text_chat text_chat
user input:
```

3.2 Test Cases

Here are some test cases for reference; users can create their own test instructions.

- Start xx tracking

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

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

  Object tracking

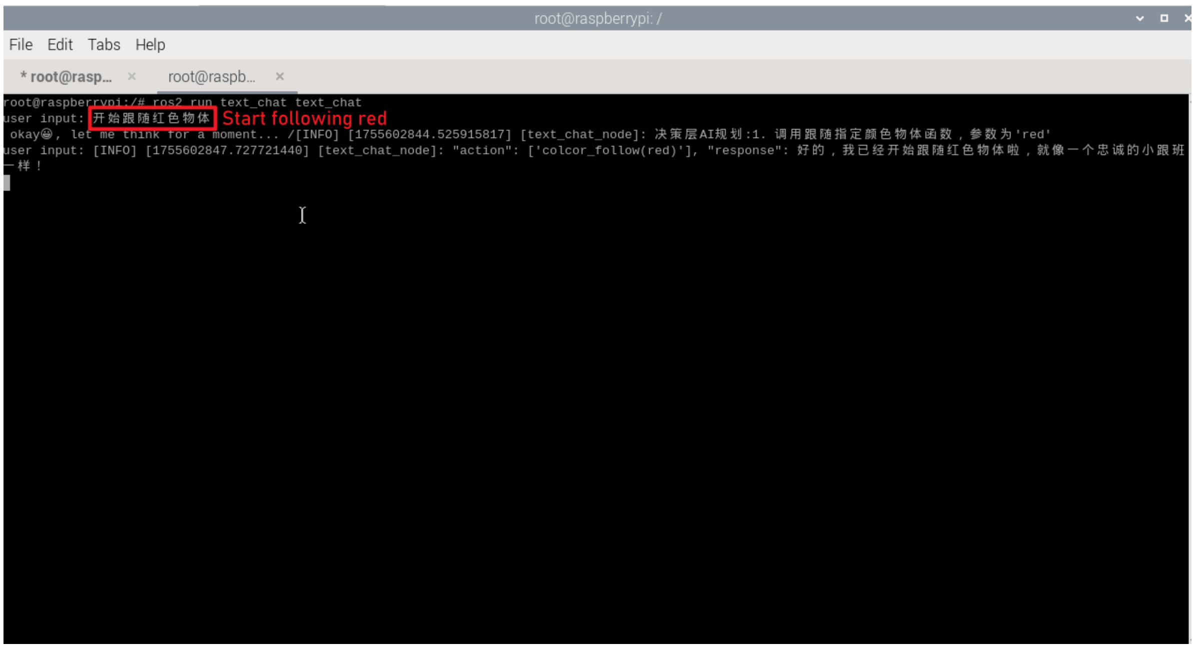
⚠ Please do not end the text with a period or any other characters!

- ⚡ Note for gimbal servo USB camera users:**No object following functionality**!
  - This tracking example uses the principle of near-increase-increase-difference. The size of the object being followed and the tracking effect are related to the recognized area. For program debugging, please refer to the single example.
  -

Following Examples	Recommended Reference Object Size
Color Following	3x3cm Building Blocks
Face Following	Face from a Mobile Phone Image
AprilTag Machine Code Following	3x3cm Building Blocks
QR Code Following	QR Code Printed on A5 Paper
Mediapipe Gesture Following	Real-person gestures
Meidpipe Human Posture Following	Real-person posture

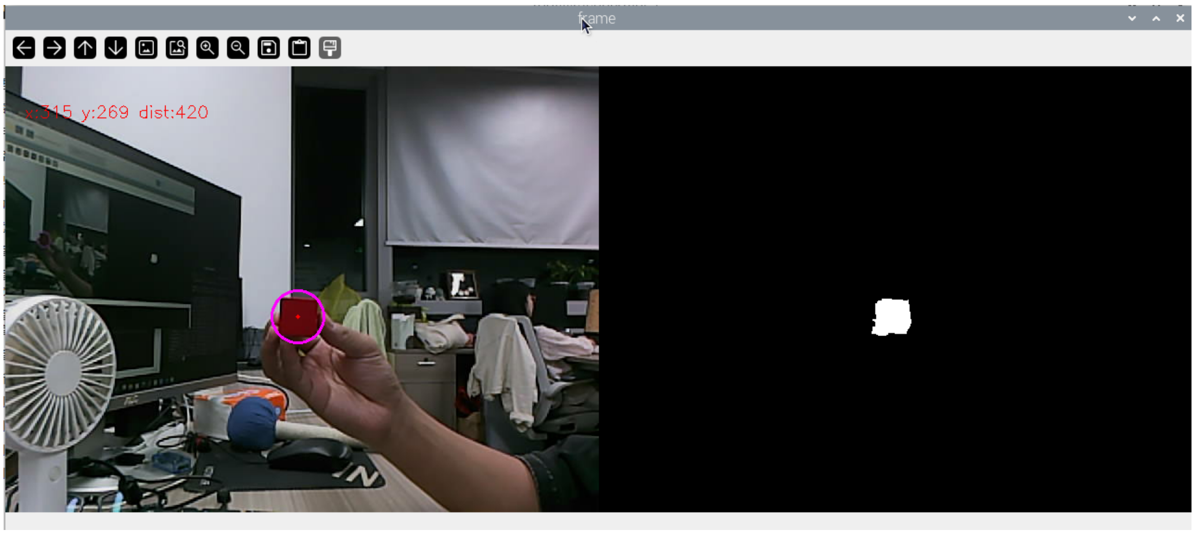
3.2.1 Example 1: "Start following red"

Type "Start following red" in the terminal. The terminal will print the following information.



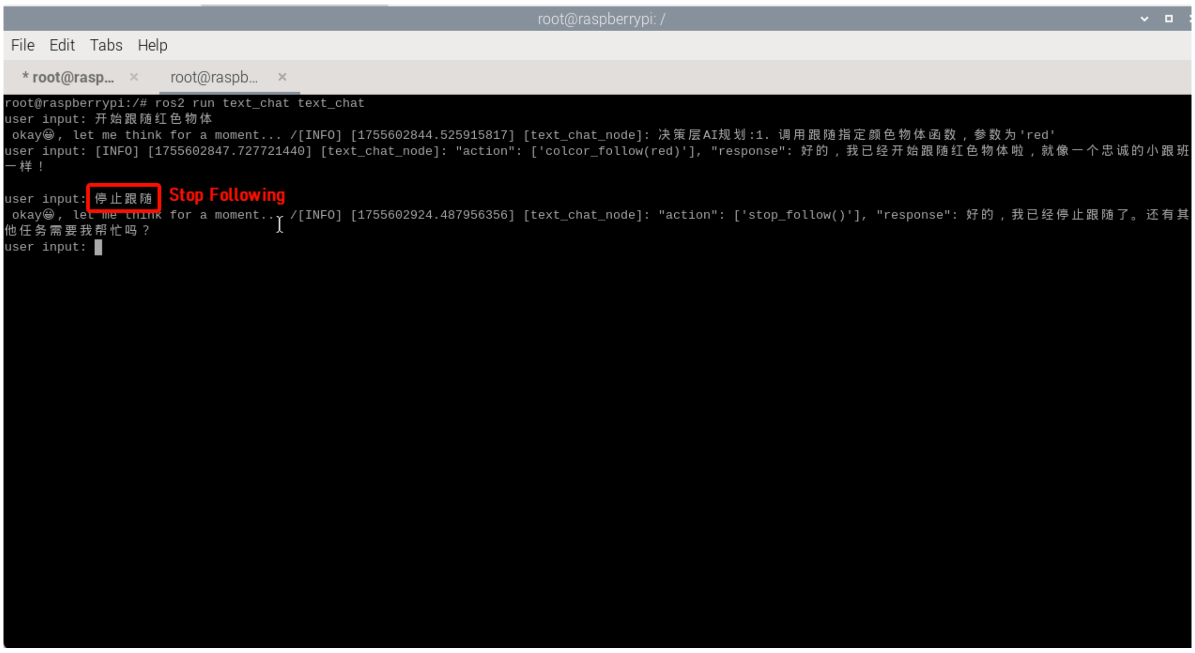
```
root@raspberrypi:/# ros2 run text_chat text_chat
user input: 开始跟随红色物体
okay😄, let me think for a moment... /[INFO] [1755602844.525915817] [text_chat_node]: 决策层AI规划:1. 调用跟随指定颜色物体函数，参数为'red'
user input: [INFO] [1755602847.727721440] [text_chat_node]: "action": ['colcor_follow(red)'], "response": 好的，我已经开始跟随红色物体啦，就像一个忠诚的小跟班一样！
```

A window titled **frame** will open on the VNC screen, displaying the current robot-viewing image.



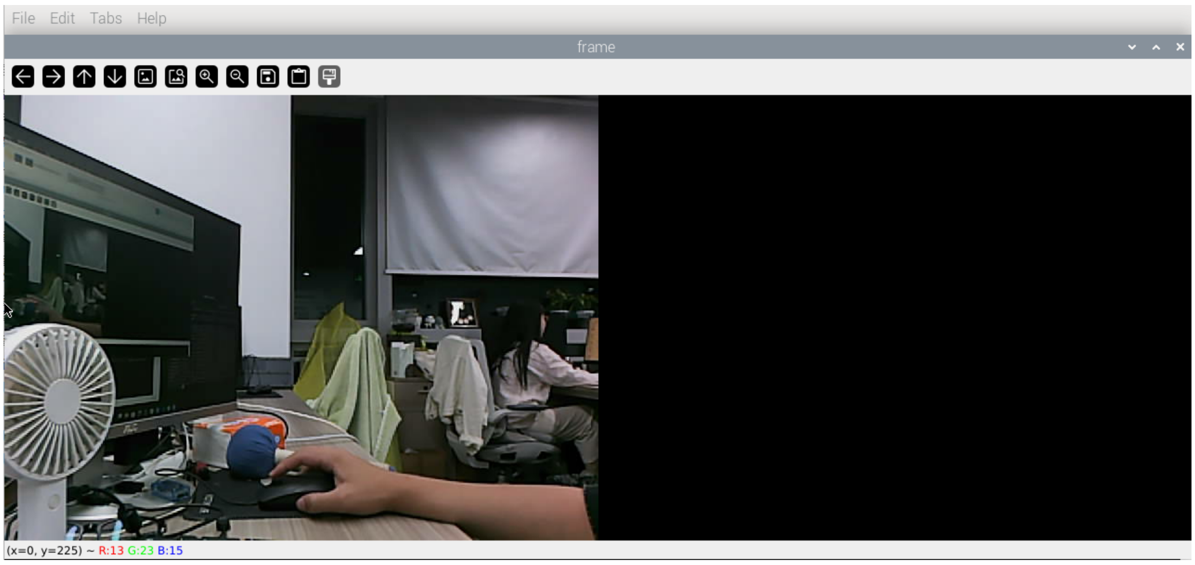
Move the object slowly, and the robot will follow.

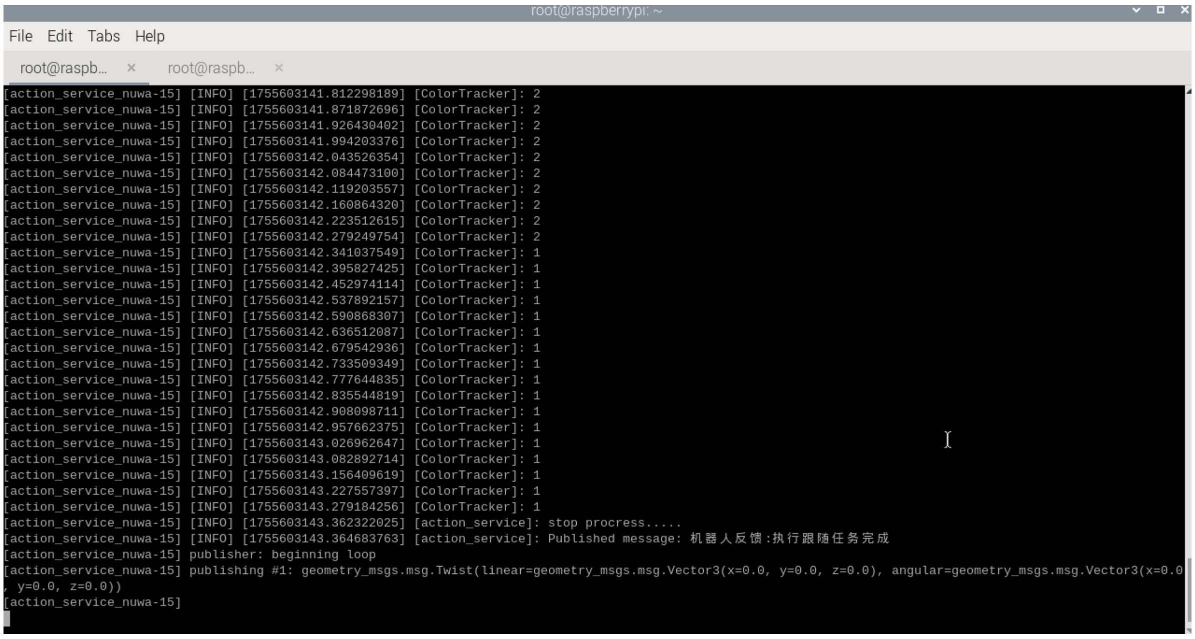
To end the task, press `ENTER` in the terminal. Press the [Enter] key to continue the conversation, enter **[Stop Following]**, or [End Following].



```
root@raspberrypi:/# ros2 run text_chat text_chat
user input: 开始跟随红色物体
 okay😀, let me think for a moment... /[INFO] [1755602844.525915817] [text_chat_node]: 决策层AI规划:1. 调用跟随指定颜色物体函数，参数为'red'
user input: [INFO] [1755602847.727721440] [text_chat_node]: "action": ['colcor_follow(red)'], "response": 好的，我已经开始跟随红色物体啦，就像一个忠诚的小跟班一样！

user input: 停止跟随
 okay😀, let me think for a moment... /[INFO] [1755602924.487956356] [text_chat_node]: "action": ['stop_follow()'], "response": 好的，我已经停止跟随了。还有其他任务需要我帮忙吗？
user input: 
```

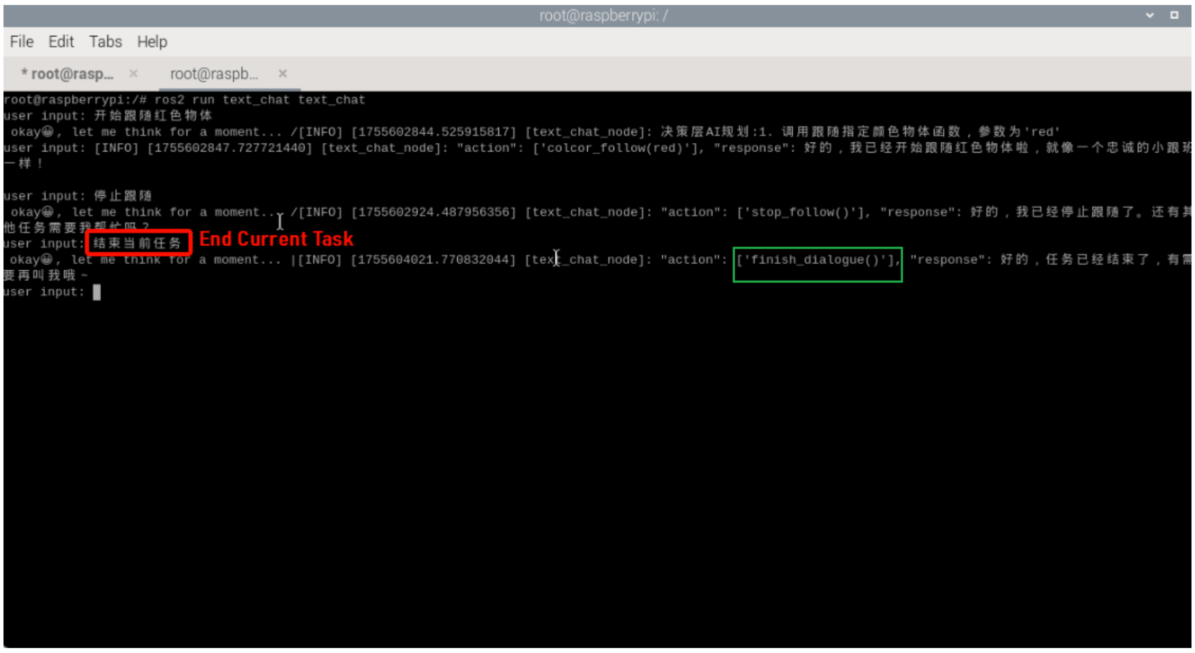
If the camera doesn't see the red object after 10 seconds, a countdown will appear on the terminal. When the countdown ends, the program will automatically stop following.



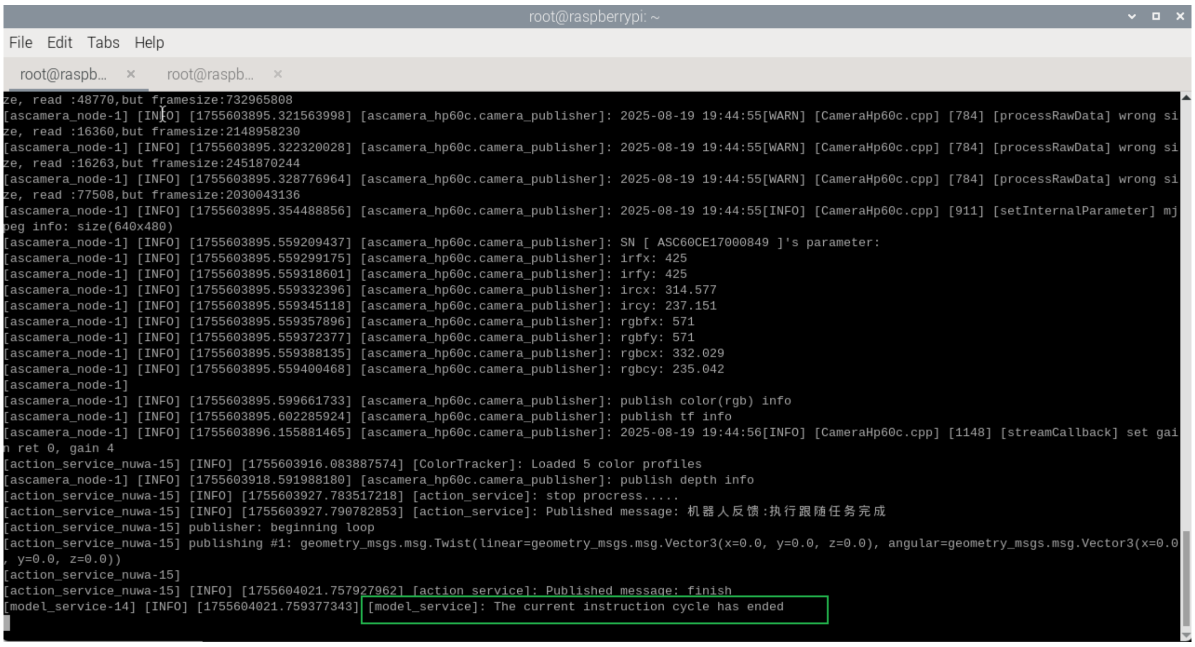


```
[action_service_nuwa-15] [INFO] [1755603141.812298189] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603141.871872696] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603141.926430402] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603141.994203376] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603142.043526354] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603142.084473100] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603142.119203557] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603142.160864320] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603142.223512615] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603142.279249754] [ColorTracker]: 2
[action_service_nuwa-15] [INFO] [1755603142.341037549] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.395827425] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.452974114] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.537892157] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.590868307] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.636512087] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.679542936] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.733509349] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.777644835] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.835544819] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.908098711] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603142.957662375] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603143.026962647] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603143.082892714] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603143.156409619] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603143.227557397] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603143.279184256] [ColorTracker]: 1
[action_service_nuwa-15] [INFO] [1755603143.362322025] [action_service]: stop procress.....
[action_service_nuwa-15] [INFO] [1755603143.364683763] [action_service]: Published message: 机器人反馈:执行跟随任务完成
[action_service_nuwa-15] publisher: beginning loop
[action_service_nuwa-15] 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_nuwa-15]
```

After completing a task, the robot will enter a waiting state. Pressing the `ENTER` key will continue the conversation. The command will be directly passed to the execution layer model, and all conversation history will be retained. We can enter the "**End Current Task**" command again to end the robot's current task cycle and start a new one.



```
root@raspberrypi:/# ros2 run text_chat text_chat
user input: 开始跟随红色物体
 okay😀, let me think for a moment... /[INFO] [1755602844.525915817] [text_chat_node]: 决策层AI规划:1. 调用跟随指定颜色物体函数，参数为'red'
user input: [INFO] [1755602847.727721440] [text_chat_node]: "action": ['colcor_follow(red)'], "response": 好的，我已经开始跟随红色物体啦，就像一个忠诚的小跟班
一样！

user input: 停止跟随
 okay😀, let me think for a moment... /[INFO] [1755602924.487956356] [text_chat_node]: "action": ['stop_follow()'], "response": 好的，我已经停止跟随了。还有其
他任务需要我帮忙吗？
user input: 结束当前任务
 okay😀, let me think for a moment... |[INFO] [1755604021.770832044] [text_chat_node]: "action": ['finish_dialogue()'], "response": 好的，任务已经结束了，有需
要再叫我哦~
user input:
```

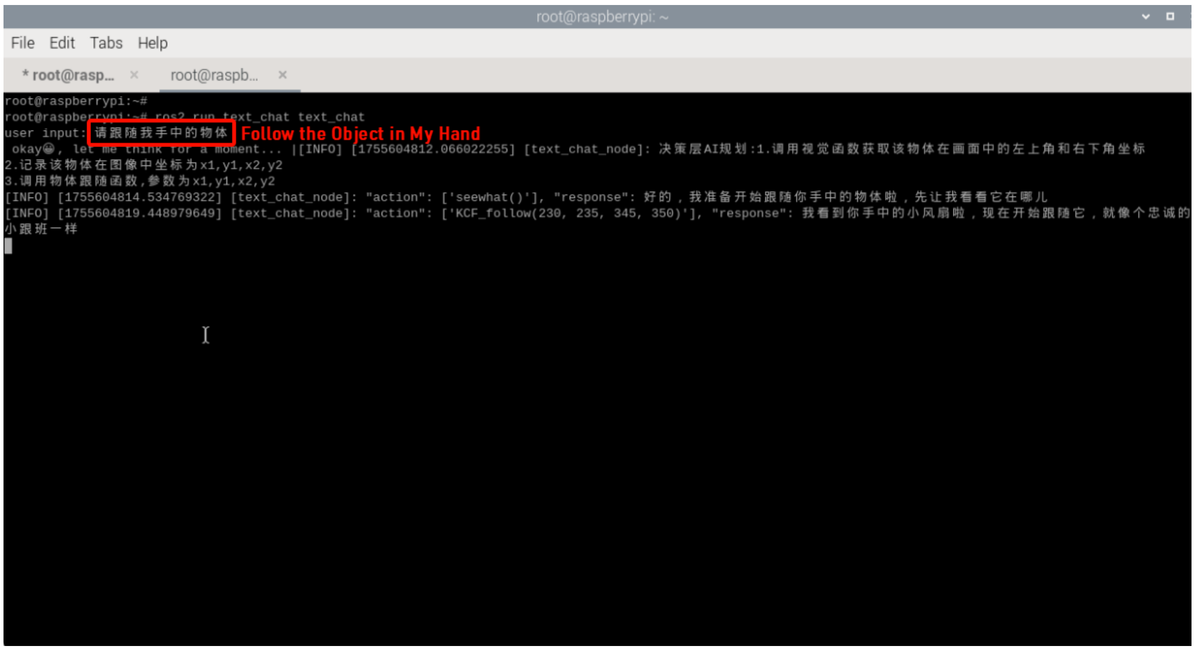


```
ze, read :48770,but framesize:732965808
[ascamera_node-1] [INFO] [1755603895.321563998] [ascamera_hp60c.camera_publisher]: 2025-08-19 19:44:55[WARN] [CameraHp60c.cpp] [784] [processRawData] wrong si
ze, read :16360,but framesize:2148958230
[ascamera_node-1] [INFO] [1755603895.322320028] [ascamera_hp60c.camera_publisher]: 2025-08-19 19:44:55[WARN] [CameraHp60c.cpp] [784] [processRawData] wrong si
ze, read :16263,but framesize:2451870244
[ascamera_node-1] [INFO] [1755603895.328776964] [ascamera_hp60c.camera_publisher]: 2025-08-19 19:44:55[WARN] [CameraHp60c.cpp] [784] [processRawData] wrong si
ze, read :77508,but framesize:2030043136
[ascamera_node-1] [INFO] [1755603895.354488856] [ascamera_hp60c.camera_publisher]: 2025-08-19 19:44:55[INFO] [CameraHp60c.cpp] [911] [setInternalParameter] mj
peg info: size(640x480)
[ascamera_node-1] [INFO] [1755603895.559209437] [ascamera_hp60c.camera_publisher]: SN [ ASC60CE17000849 ]'s parameter:
[ascamera_node-1] [INFO] [1755603895.559299175] [ascamera_hp60c.camera_publisher]: irfx: 425
[ascamera_node-1] [INFO] [1755603895.559318601] [ascamera_hp60c.camera_publisher]: irfy: 425
[ascamera_node-1] [INFO] [1755603895.559332396] [ascamera_hp60c.camera_publisher]: ircx: 314.577
[ascamera_node-1] [INFO] [1755603895.559345118] [ascamera_hp60c.camera_publisher]: ircy: 237.151
[ascamera_node-1] [INFO] [1755603895.559357896] [ascamera_hp60c.camera_publisher]: rgbfx: 571
[ascamera_node-1] [INFO] [1755603895.559372377] [ascamera_hp60c.camera_publisher]: rgbfy: 571
[ascamera_node-1] [INFO] [1755603895.559388135] [ascamera_hp60c.camera_publisher]: rgbcx: 332.029
[ascamera_node-1] [INFO] [1755603895.559400468] [ascamera_hp60c.camera_publisher]: rgbcy: 235.042
[ascamera_node-1]
[ascamera_node-1] [INFO] [1755603895.599661733] [ascamera_hp60c.camera_publisher]: publish color(rgb) info
[ascamera_node-1] [INFO] [1755603895.602285924] [ascamera_hp60c.camera_publisher]: publish tf info
[ascamera_node-1] [INFO] [1755603896.155881465] [ascamera_hp60c.camera_publisher]: 2025-08-19 19:44:56[INFO] [CameraHp60c.cpp] [1148] [streamCallback] set gai
n ret 0, gain 4
[action_service_nuwa-15] [INFO] [1755603916.083887574] [ColorTracker]: Loaded 5 color profiles
[ascamera_node-1] [INFO] [1755603918.591988180] [ascamera_hp60c.camera_publisher]: publish depth info
[action_service_nuwa-15] [INFO] [1755603927.783517218] [action_service]: stop procress.....
[action_service_nuwa-15] [INFO] [1755603927.790782853] [action_service]: Published message: 机器人反馈:执行跟随任务完成
[action_service_nuwa-15] publisher: beginning loop
[action_service_nuwa-15] 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_nuwa-15]
[action_service_nuwa-15] [INFO] [1755604021.757927962] [action_service]: Published message: finish
[model_service-14] [INFO] [1755604021.759377343] [model_service]: The current instruction cycle has ended
```

3.2.2 Example 2: "Follow the Object in My Hand" (Depth Camera)

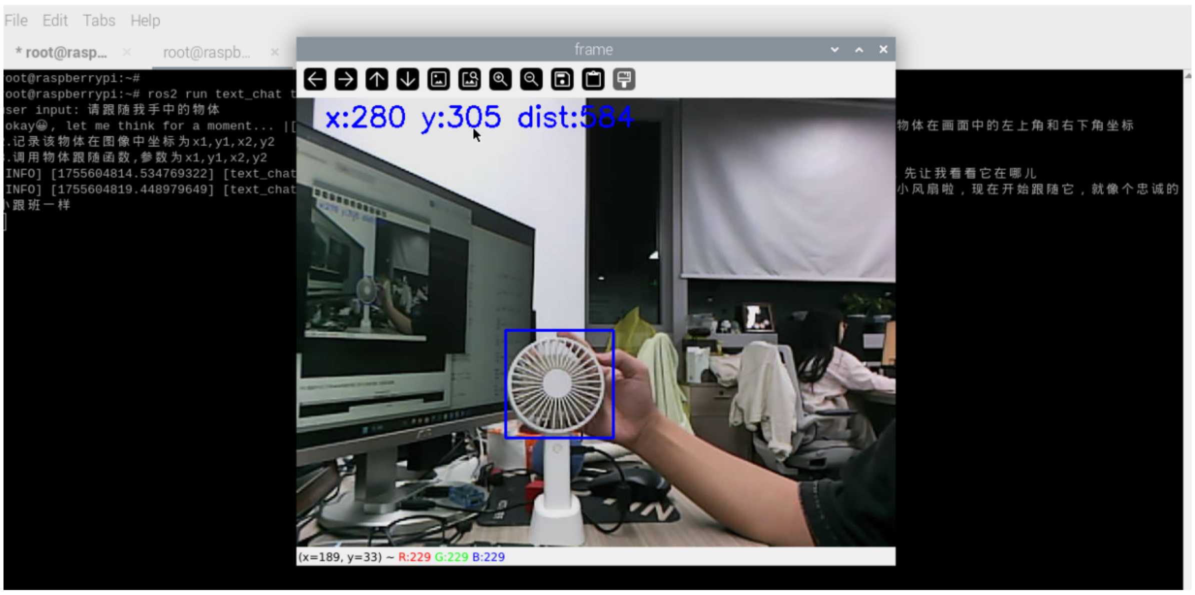
⚠ The coordinates obtained by following the object in this example are entirely derived from the inference of the large AI model. Therefore, it is recommended to use a newer model for better results!

Enter "Please follow the object in my hand" in the terminal. The terminal will print the following information:



```
root@raspberrypi:~#
root@raspberrypi:~# ros2 run text_chat text_chat
user input:请跟随我手中的物体
 okay😀, let me think for a moment... |[INFO] [1755604812.066022255] [text_chat_node]: 决策层AI规划:1.调用视觉函数获取该物体在画面中的左上角和右下角坐标
2.记录该物体在图像中坐标为x1,y1,x2,y2
3.调用物体跟随函数,参数为x1,y1,x2,y2
[INFO] [1755604814.534769322] [text_chat_node]: "action": ['seewhat()'], "response": 好的，我准备开始跟随你手中的物体啦，先让我看看它在哪儿
[INFO] [1755604819.448979649] [text_chat_node]: "action": ['KCF_follow(230, 235, 345, 350)'], "response": 我看到你手中的小风扇啦，现在开始跟随它，就像个忠诚的
小跟班一样
```

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

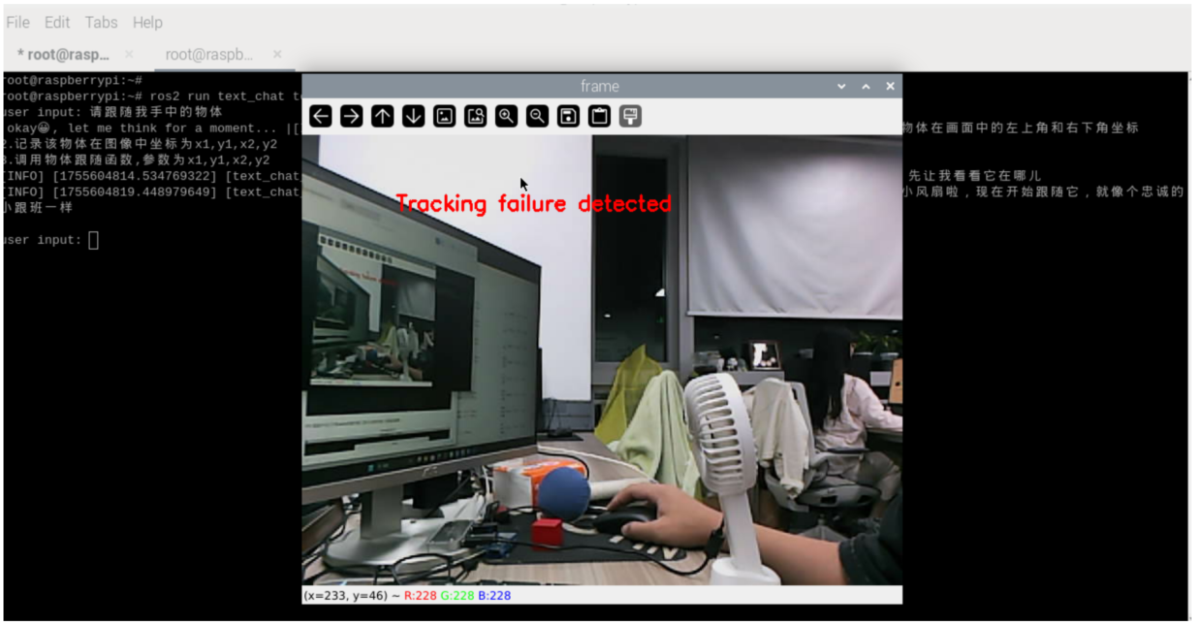

File Edit Tabs Help
* root@rasp... root@raspb...
frame
x:280 y:305 dist:584
(x=189, y=33) ~ R:229 G:229 B:229

```
oot@raspberrypi:~#
oot@raspberrypi:~# ros2 run text_chat t
ser input: 请跟随我手中的物体
okay😀, let me think for a moment... |[
.记录该物体在图像中坐标为x1,y1,x2,y2
.调用物体跟随函数,参数为x1,y1,x2,y2
INFO] [1755604814.534769322] [text_chat
INFO] [1755604819.448979649] [text_chat
小跟班一样
```

```
物体在画面中的左上角和右下角坐标

先让我看看它在哪儿
小风扇啦，现在开始跟随它，就像个忠诚的
```

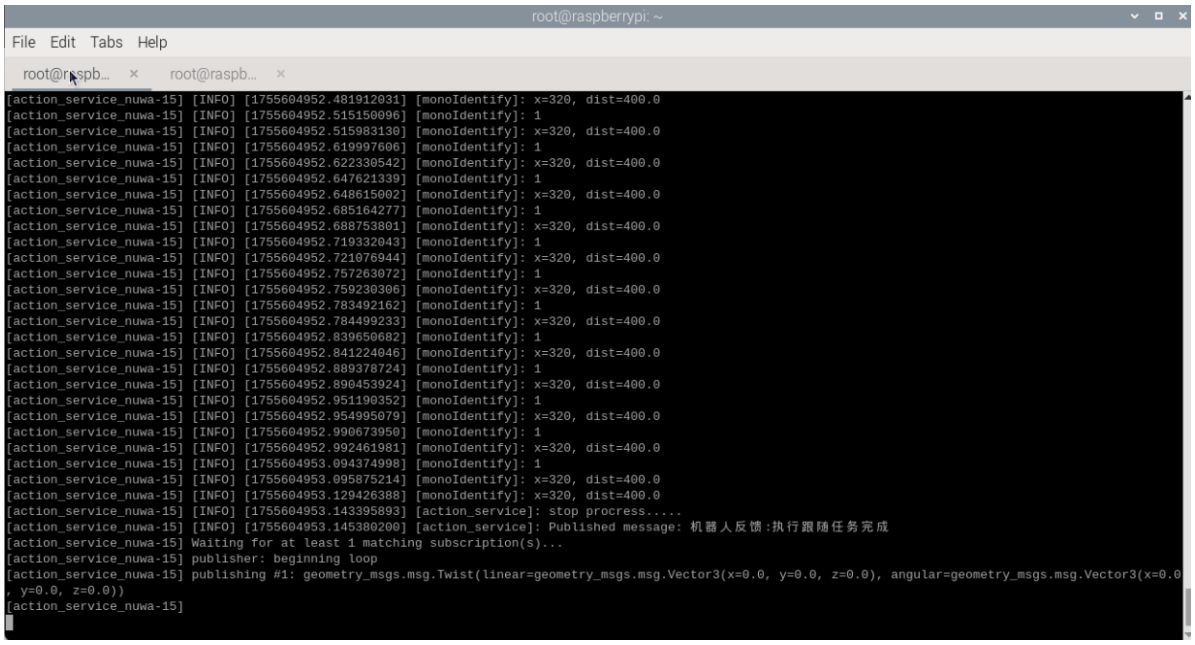
Move the object slowly, and the robot will follow.

If the screen If there are no targets to follow, 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 that the task has been completed.


File Edit Tabs Help
* root@rasp... root@raspb...
frame
Tracking failure detected
(x=233, y=46) ~ R:228 G:228 B:228

```
oot@raspberrypi:~#
oot@raspberrypi:~# ros2 run text_chat t
ser input: 请跟随我手中的物体
okay😀, let me think for a moment... |[
.记录该物体在图像中坐标为x1,y1,x2,y2
.调用物体跟随函数,参数为x1,y1,x2,y2
INFO] [1755604814.534769322] [text_chat
INFO] [1755604819.448979649] [text_chat
小跟班一样

ser input:
```

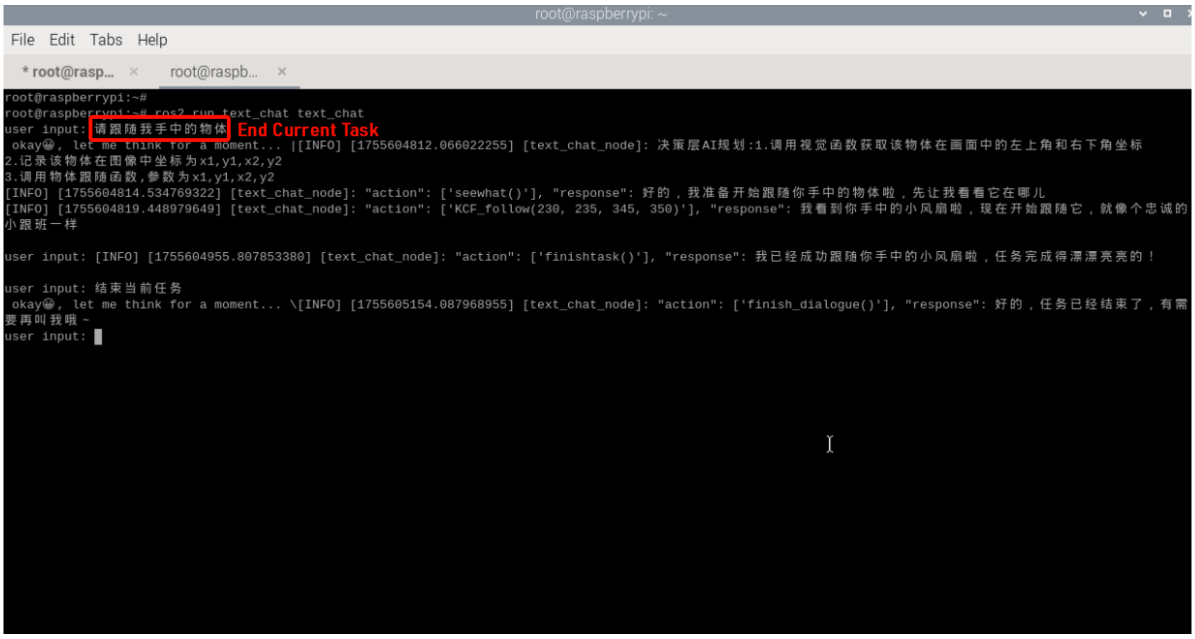
```
物体在画面中的左上角和右下角坐标

先让我看看它在哪儿
小风扇啦，现在开始跟随它，就像个忠诚的
```



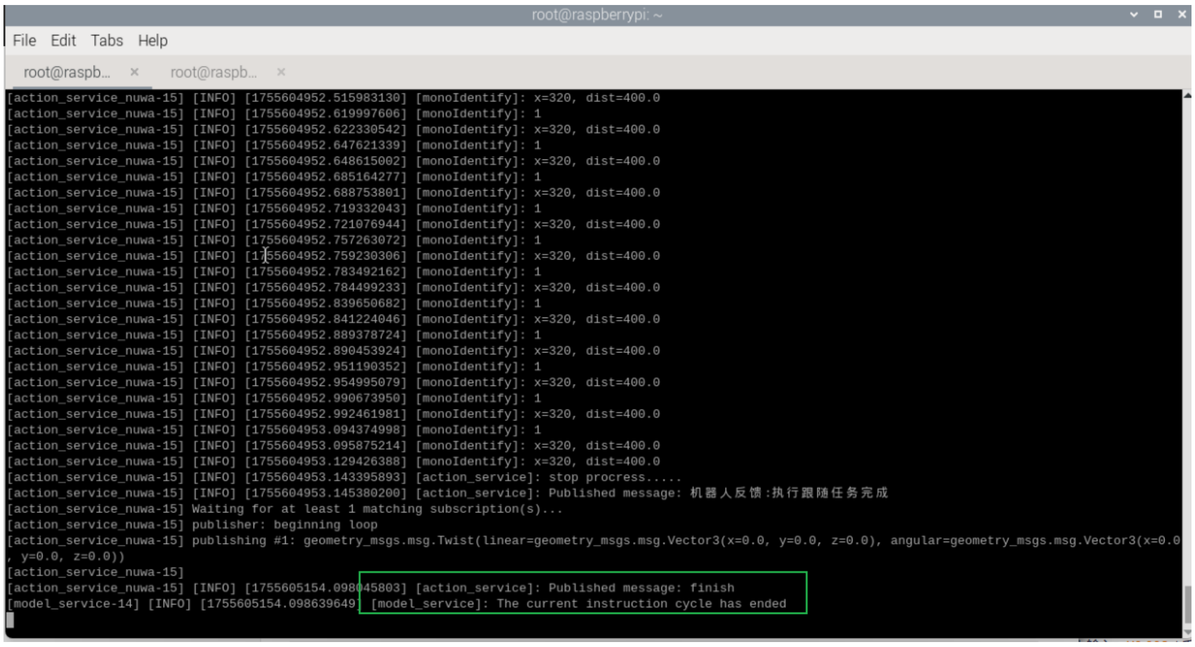
```
[action_service_nuwa-15] [INFO] [1755604952.481912031] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.515150096] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.515983130] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.619997606] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.622330542] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.647621339] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.648615002] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.685164277] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.688753801] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.719332043] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.721076944] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.757263072] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.759230306] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.783492162] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.784499233] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.839650682] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.841224046] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.889378724] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.890453924] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.951190352] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.954995079] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.990673950] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.992461981] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604953.094374998] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604953.095875214] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604953.129426388] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604953.143395893] [action_service]: stop procress.....
[action_service_nuwa-15] [INFO] [1755604953.145380200] [action_service]: Published message: 机器人反馈:执行跟随任务完成
[action_service_nuwa-15] Waiting for at least 1 matching subscription(s)...
[action_service_nuwa-15] publisher: beginning loop
[action_service_nuwa-15] 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_nuwa-15]
```

To manually end the task, press the `ENTER` key in the terminal and continue the conversation, entering either [Stop Following] or [End Following].

After completing a task, the robot enters a waiting state. Commands at this point are directly passed to the execution layer model, and all conversation history is retained. You can enter the "**End Current Task**" command again to end the current task cycle and start a new one.



```
root@raspberrypi:~#
root@raspberrypi:~# ros2 run text_chat text_chat
user input: 请跟随我手中的物体
 okay😀, let me think for a moment... |[INFO] [1755604812.066022255] [text_chat_node]: 决策层AI规划:1.调用视觉函数获取该物体在画面中的左上角和右下角坐标
2.记录该物体在图像中坐标为x1,y1,x2,y2
3.调用物体跟随函数,参数为x1,y1,x2,y2
[INFO] [1755604814.534769322] [text_chat_node]: "action": ['seewhat()'], "response": 好的，我准备开始跟随你手中的物体啦，先让我看看它在哪儿
[INFO] [1755604819.448979649] [text_chat_node]: "action": ['KCF_follow(230, 235, 345, 350)'], "response": 我看到你手中的小风扇啦，现在开始跟随它，就像个忠诚的
小跟班一样

user input: [INFO] [1755604955.807853380] [text_chat_node]: "action": ['finishtask()'], "response": 我已经成功跟随你手中的小风扇啦，任务完成得漂漂亮亮的！

user input: 结束当前任务
 okay😀, let me think for a moment... \[INFO] [1755605154.087968955] [text_chat_node]: "action": ['finish_dialogue()'], "response": 好的，任务已经结束了，有需
要再叫我哦~
user input: 
```



```
[action_service_nuwa-15] [INFO] [1755604952.515983130] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.619997606] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.622330542] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.647621339] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.648615002] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.685164277] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.688753801] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.719332043] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.721076944] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.757263072] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.759230306] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.783492162] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.784499233] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.839650682] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.841224046] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.889378724] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.890453924] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.951190352] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.954995079] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604952.990673950] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604952.992461981] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604953.094374998] [monoIdentify]: 1
[action_service_nuwa-15] [INFO] [1755604953.095875214] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604953.129426388] [monoIdentify]: x=320, dist=400.0
[action_service_nuwa-15] [INFO] [1755604953.143395893] [action_service]: stop procress.....
[action_service_nuwa-15] [INFO] [1755604953.145380200] [action_service]: Published message: 机器人反馈:执行跟随任务完成
[action_service_nuwa-15] Waiting for at least 1 matching subscription(s)...
[action_service_nuwa-15] publisher: beginning loop
[action_service_nuwa-15] 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_nuwa-15]
[action_service_nuwa-15] [INFO] [1755605154.098045803] [action_service]: Published message: finish
[model_service-14] [INFO] [1755605154.098639649] [model_service]: The current instruction cycle has ended
```

4. Source Code Parsing

Source code is located at:

Jetson Orin Nano:

```
#NUWA camera user
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_s
ervice_nuwa.py
#USB camera user
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/action_s
ervice_usb.py
```

```
/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/model_se
rvice.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
```

```
/home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/model_s
ervice.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
```

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/largemodel/largemodel/model_service.p
y
```

4.1 Example 1

4.1.1 Nuwa Depth Camera, action_service_nuwa.py

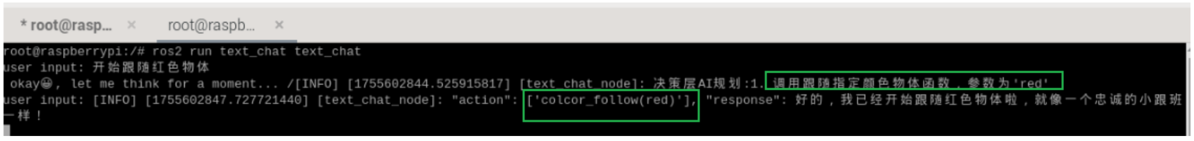
Example 1 uses the **seewhat**, **colcor_follow**, and **stop_follow()** methods in the **CustomActionServer** class.

The **seewhat** function primarily retrieves the color image from the depth camera.

The **colcor_follow(self, color)** function performs color following.

The **stop_follow()** function issues a stop command to follow.

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



```
root@raspberrypi:/# ros2 run text_chat text_chat
user input: 开始跟随红色物体
 okay😀, let me think for a moment... /[INFO] [1755602844.525915817] [text_chat_node]: 决策层AI规划:1. 调用跟随指定颜色物体函数，参数为'red'
user input: [INFO] [1755602847.727721440] [text_chat_node]: "action": ['colcor_follow(red)'], "response": 好的，我已经开始跟随红色物体啦，就像一个忠诚的小跟班一样！
```

```
#Start the color line patrol subprocess
    process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl_depth',
'voice_colorTracker','--ros-args','-p',f'colcor:={target_color}'])
```

The source code path of the subprocess in the program,

`~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl_depth/yahboomcar_voice_ctrl_depth/follow/voice_colorTracker.py`

```python
def colcor_follow(self,color):
    self.colcor_follow_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

    #Start the color line patrol subprocess
    process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl_depth',
'voice_colorTracker','--ros-args','-p',f'colcor:={target_color}'])

    #Waiting to stop following instructions
    while not self.colcor_follow_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 target being followed is lost for more than 10 seconds,

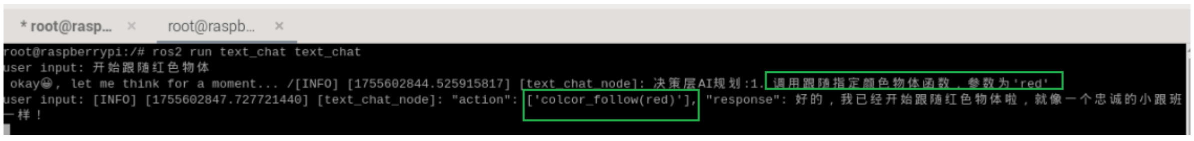
the **stop_follow** method is called, sending the future.done signal. The `while not self.colcor_follow_future.done()` block in the **colcor_follow** function then exits. The **kill_process_tree** method is then called to recursively kill the child process tree. Finally, the status of the execution action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

4.1.2 USB Camera, action_service_usb.py

Example 1 uses the **seewhat**, **colorFollow**, and **stop_track()** methods in the **CustomActionServer** class.

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

This section focuses on the **colorFollow(self, color)** function, which requires a color parameter (red, green, blue, and yellow).



```
root@raspberrypi:/# ros2 run text_chat text_chat
user input: 开始跟随红色物体
 okay😀, let me think for a moment... /[INFO] [1755602844.525915817] [text_chat_node]: 决策层AI规划:1. 调用跟随指定颜色物体函数，参数为'red'
user input: [INFO] [1755602847.727721440] [text_chat_node]: "action": ['colcor_follow(red)'], "response": 好的，我已经开始跟随红色物体啦，就像一个忠诚的小跟班
一样！
```

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

The source code path of the subprocess in the program,

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

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

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_track** method is called, sending the future.done signal. The `while not self.colorFollow_future.done()` block in the **colorFollow** 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.

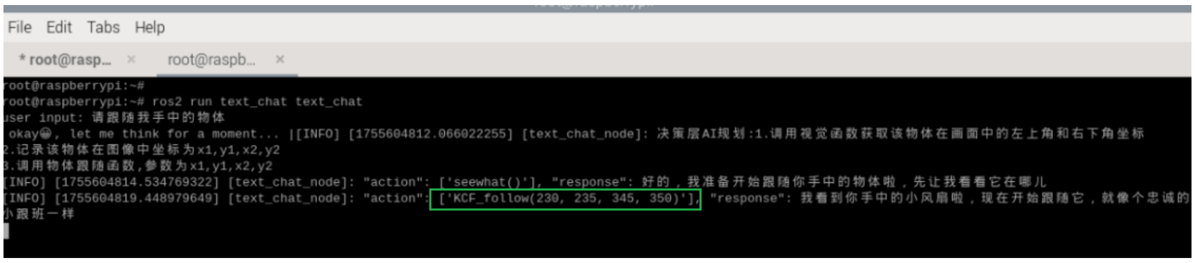
4.2 Example 2

4.2.1 nuwa Depth Camera, action_service_nuwa.py

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

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

The **seewhat** function primarily retrieves the camera's color image. The **KCF_follow(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 (230, 345) and the lower-right corner is (235, 350).



```
root@raspberrypi:~#
root@raspberrypi:~# ros2 run text_chat text_chat
user input: 请跟随我手中的物体
 okay😀, let me think for a moment... |[INFO] [1755604812.066022255] [text_chat_node]: 决策层AI规划:1.调用视觉函数获取该物体在画面中的左上角和右下角坐标
2.记录该物体在图像中坐标为x1,y1,x2,y2
3.调用物体跟随函数,参数为x1,y1,x2,y2
[INFO] [1755604814.534769322] [text_chat_node]: "action": ['seewhat()'], "response": 好的，我准备开始跟随你手中的物体啦，先让我看看它在哪儿
[INFO] [1755604819.448979649] [text_chat_node]: "action": ['KCF_follow(230, 235, 345, 350)'], "response": 我看到你手中的小风扇啦，现在开始跟随它，就像个忠诚的小跟班一样
```

```
#Start the object tracking subprocess
process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl_depth', 'voice_KCF_Tracker','--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_depth/yahboomcar_voice_ctrl_depth/kcf/voice_KCF_Tracker.py`

```
def KCF_follow(self,x1,y1,x2,y2):
    self.KCF_follow_future = Future() #Reset Future object
    x1 = int(x1)
    y1 = int(y1)
    x2 = int(x2)
    y2 = int(y2)

    process_1 = subprocess.Popen(['ros2', 'run', 'yahboomcar_voice_ctrl_depth', 'voice_KCF_Tracker','--ros-args','-p',f'x1:={x1}','-p',f'y1:={y1}','-p',f'x2:={x2}','-p',f'y2:={y2}'])
    # time.sleep(1.0)#Sleep for 2 seconds to wait for the thread to stabilize

    while not self.KCF_follow_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 target is lost for more than 10 seconds,

the **stop_follow** method is called, sending the future.done signal. The `while not self.KCF_follow_future.done()` block in the **KCF_follow** 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.