

# Multimodal visual understand+robotic arm grab

---

## 1. Course Content

---

1. Learning to use the robot's visual understanding combined with the robot's gripping capabilities
2. Analyze the newly emerged key source code
3. Note: For the same test command, the big model's responses will not be exactly the same and will be slightly different 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 [2. AI Big Model Basics - 5. Configuring the AI Big Model] course.

## 2. Preparation

---

### 2.1 Content Description

This course uses the Jetson Orin NX 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 course in the terminal. For instructions on entering the Docker container from the host computer, refer to the **[Configuration and Operation Guide] -- [Enter the Docker (Jetson Nano and Raspberry Pi 5 users see here)]** section of this product tutorial. For Orin and NX boards, simply open a terminal and enter the commands mentioned in this course.

### 2.2 Start the Agent

**Note: To test all cases, you must start the docker agent first. If it has already been started, you do not need to start it again.**

Enter the command in the vehicle terminal:

```
sh start_agent.sh
```

The terminal prints the following information, indicating that the connection is successful

```

jetson@yahboom: ~
jetson@yahboom: - 177x26

[jetson@yahboom: ~]$ ./open_agent.sh
[1749538760_063253] [INFO] | TermiosAgentLinux.cpp | init | running... | fd: 3
[1749538760_064012] [INFO] | Root.cpp | set_verbose_level | verbose_level: 4
[1749538760_941420] [INFO] | Root.cpp | create_client | create | client_key: 0x0DA4EFC, session_id: 0x81
[1749538760_941537] [INFO] | SessionManager.hpp | establish_session | session established | client_key: 0x0DA4EFC, address: 0
[1749538760_079971] [INFO] | ProxyClient.cpp | create_participant | participant created | client_key: 0x0DA4EFC, participant_id: 0x000(1)
[1749538760_983835] [INFO] | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA4EFC, topic_id: 0x000(2), participant_id: 0x000(1)
[1749538760_988244] [INFO] | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA4EFC, publisher_id: 0x000(3), participant_id: 0x000(1)
[1749538760_993117] [INFO] | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA4EFC, datawriter_id: 0x000(5), publisher_id: 0x000(3)
[1749538760_997675] [INFO] | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA4EFC, topic_id: 0x001(2), participant_id: 0x000(1)
[1749538761_001121] [INFO] | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA4EFC, publisher_id: 0x001(3), participant_id: 0x000(1)
[1749538761_007277] [INFO] | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA4EFC, datawriter_id: 0x001(5), publisher_id: 0x000(3)
[1749538761_010634] [INFO] | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA4EFC, topic_id: 0x002(2), participant_id: 0x000(1)
[1749538761_014290] [INFO] | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA4EFC, publisher_id: 0x002(3), participant_id: 0x000(1)
[1749538761_020171] [INFO] | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA4EFC, datawriter_id: 0x002(5), publisher_id: 0x002(3)
[1749538761_023939] [INFO] | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA4EFC, topic_id: 0x003(2), participant_id: 0x000(1)
[1749538761_029173] [INFO] | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA4EFC, publisher_id: 0x003(3), participant_id: 0x000(1)
[1749538761_034377] [INFO] | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA4EFC, datawriter_id: 0x003(5), publisher_id: 0x003(3)
[1749538761_038946] [INFO] | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA4EFC, topic_id: 0x004(2), participant_id: 0x000(1)
[1749538761_042215] [INFO] | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA4EFC, publisher_id: 0x004(3), participant_id: 0x000(1)
[1749538761_048422] [INFO] | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA4EFC, datawriter_id: 0x004(5), publisher_id: 0x004(3)
[1749538761_051660] [INFO] | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA4EFC, topic_id: 0x005(2), participant_id: 0x000(1)
[1749538761_057494] [INFO] | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA4EFC, publisher_id: 0x005(3), participant_id: 0x000(1)
[1749538761_062183] [INFO] | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA4EFC, datawriter_id: 0x005(5), publisher_id: 0x005(3)
[1749538761_066842] [INFO] | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA4EFC, topic_id: 0x006(2), participant_id: 0x000(1)
[1749538761_071217] [INFO] | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x0DA4EFC, subscriber_id: 0x000(4), participant_id: 0x000(1)

```

## 3. Run the case

### 3.1 Startup Program

Connect to the vehicle computer screen via VNC, open the terminal and enter the command:

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

Wait for the initialization program to complete, as shown below:

```

jetson@yahboom: ~
[component_container-1] [INFO] [1749289425.553596031] [camera.camera]: Disable frame sync
[component_container-1] [INFO] [1749289425.553801377] [camera.camera]: Device DaBai DCW2 connected
[component_container-1] [INFO] [1749289425.553823041] [camera.camera]: Serial number: AUIMB4200AW
[component_container-1] [INFO] [1749289425.553839169] [camera.camera]: Firmware version: RD1014
[component_container-1] [INFO] [1749289425.553855233] [camera.camera]: Hardware version:
[component_container-1] [INFO] [1749289425.553865377] [camera.camera]: device unique id: 1-2.3.1-9
[component_container-1] [INFO] [1749289425.553889057] [camera.camera]: Current node pid: 1079135
[component_container-1] [INFO] [1749289425.553899553] [camera.camera]: usb connect type: USB2.0
[component_container-1] [INFO] [1749289425.553908449] [camera.camera]: Start device cost 1291 ms
[component_container-1] [INFO] [1749289425.553918466] [camera.camera]: Initialize device cost 949 ms
[component_container-1] [INFO] [1749289425.891162561] [camera.camera]: Publishing static transform from
ir to depth
[component_container-1] [INFO] [1749289425.891285986] [camera.camera]: Translation 0, 0, 0
[component_container-1] [INFO] [1749289425.891304322] [camera.camera]: Rotation 0, 0, 0, 1
[component_container-1] [INFO] [1749289425.891322882] [camera.camera]: Publishing static transform from
color to depth
[component_container-1] [INFO] [1749289425.891345346] [camera.camera]: Translation 12.317, 0.046452, 1.6
3189
[component_container-1] [INFO] [1749289425.891362018] [camera.camera]: Rotation -0.00113323, 0.00116674,
0.000693509, 0.999998
[component_container-1] [INFO] [1749289425.891377666] [camera.camera]: Publishing static transform from
depth to depth
[component_container-1] [INFO] [1749289425.891391938] [camera.camera]: Translation 0, 0, 0
[component_container-1] [INFO] [1749289425.891512963] [camera.camera]: Rotation 0, 0, 0, 1
[asr-5] [INFO] [1749289435.107467316] [asr]: The online asr model :parafomer-realtime-v2 is loaded
[asr-5] [INFO] [1749289435.121443184] [asr]: asr_node Initialization completed
[model_service-3] [INFO] [1749289435.708789083] [model_service]: LargeModelService node Initialization c
ompleted...

```

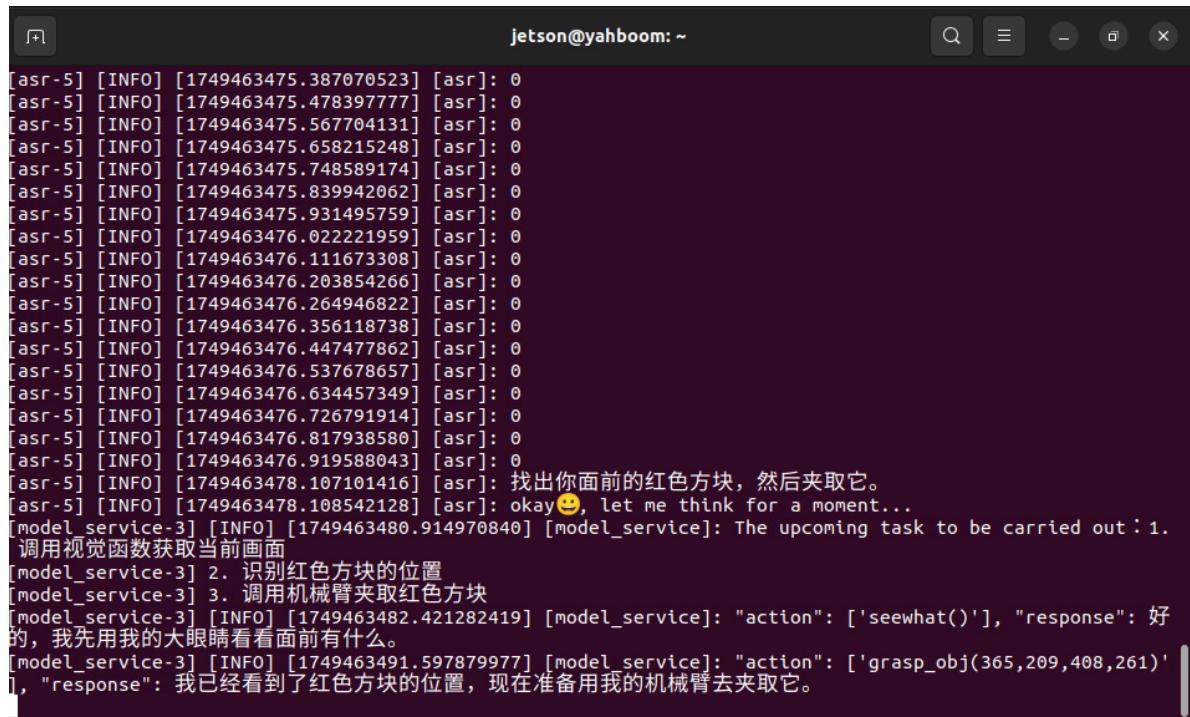
## 3.2 Test Cases

Here are two test cases. Users can compile their own test instructions.

- Find the red block in front of you and grab it
- Place the red block in front of you to the right of the blue block
- Please help me remove the machine code in front of you that is higher than 5 cm
- Tracking machine number three

### 3.2.1 Example 1: "Find the red block in front of you and grab it"

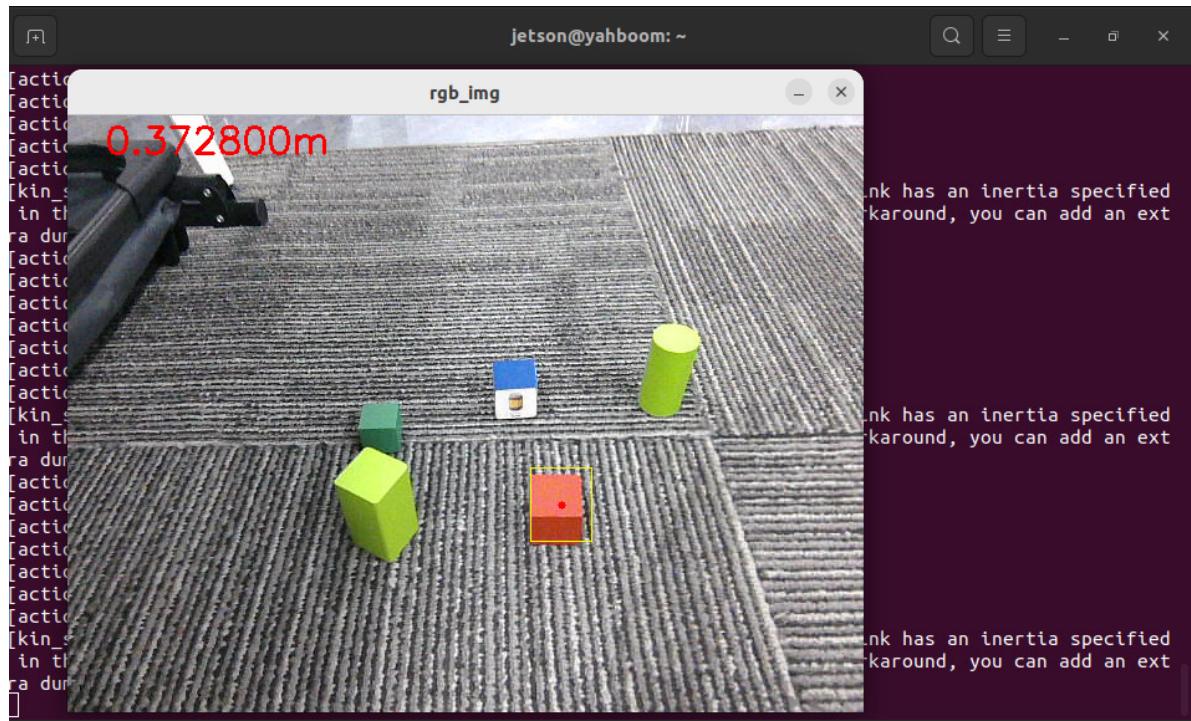
First, use "Hi, yahboom" to wake up the robot. The robot responds: "I'm here, please tell me what to do." After the robot responds, the buzzer beeps briefly (beep—). The user can speak and wait for the robot to reply after speaking. The terminal prints the following information:



The screenshot shows a terminal window titled "jetson@yahboom: ~". The log output is as follows:

```
[asr-5] [INFO] [1749463475.387070523] [asr]: 0
[asr-5] [INFO] [1749463475.478397777] [asr]: 0
[asr-5] [INFO] [1749463475.567704131] [asr]: 0
[asr-5] [INFO] [1749463475.658215248] [asr]: 0
[asr-5] [INFO] [1749463475.748589174] [asr]: 0
[asr-5] [INFO] [1749463475.839942062] [asr]: 0
[asr-5] [INFO] [1749463475.931495759] [asr]: 0
[asr-5] [INFO] [1749463476.022221959] [asr]: 0
[asr-5] [INFO] [1749463476.111673308] [asr]: 0
[asr-5] [INFO] [1749463476.203854266] [asr]: 0
[asr-5] [INFO] [1749463476.264946822] [asr]: 0
[asr-5] [INFO] [1749463476.356118738] [asr]: 0
[asr-5] [INFO] [1749463476.447477862] [asr]: 0
[asr-5] [INFO] [1749463476.537678657] [asr]: 0
[asr-5] [INFO] [1749463476.634457349] [asr]: 0
[asr-5] [INFO] [1749463476.726791914] [asr]: 0
[asr-5] [INFO] [1749463476.817938580] [asr]: 0
[asr-5] [INFO] [1749463476.919588043] [asr]: 0
[asr-5] [INFO] [1749463478.107101416] [asr]: 找出你面前的红色方块，然后来取它。
[asr-5] [INFO] [1749463478.108542128] [asr]: okay😊, let me think for a moment...
[model_service-3] [INFO] [1749463480.914970840] [model_service]: The upcoming task to be carried out: 1.
调用视觉函数获取当前画面
[model_service-3] 2. 识别红色方块的位置
[model_service-3] 3. 调用机械臂夹取红色方块
[model_service-3] [INFO] [1749463482.421282419] [model_service]: "action": ["seewhat()"], "response": 好的，我先用我的大眼睛看看面前有什么。
[model_service-3] [INFO] [1749463491.597879977] [model_service]: "action": ["grasp_obj(365,209,408,261)"], "response": 我已经看到了红色方块的位置，现在准备用我的机械臂去夹取它。
```

`grasp_obj()` After the function is called, a window titled `rgb_img` will open on the VNC screen, displaying the image from the robot's current perspective. The robot will automatically adjust the distance to the target object. After the distance adjustment is complete, the robot will use its robotic arm to grasp the target object.



After the test is completed, wake up the robot again to let it end the current task.

Note: After the robot grabs an object, the robotic arm will be in a maintenance state. If you need the robotic arm to return to its initial state, you can use the following method

- Method 1: Wake up the robot again and let it put down the red block it just grabbed
- Method 2: Wake up the robot again and issue the command "End current task" to let the robot end the current task. After the task cycle is completed, the robot arm will reset to its initial posture.

### 3.2.2 Case 2: "Place the red block in front of you to the right of the blue block"

After waking up the robot, say the command "Place the red square in front of you to the right of the blue square." The terminal prints the following information. Based on the output of the decision-making model, the robot's strategy for completing the task is: 1. Identify the image—2. Locate the red square—3. Grab the red square—4. Move it to the right—5. Place the square.

```

jetson@yahboom: ~
[asr-5] [INFO] [1749468348.171435630] [asr]: 0
[asr-5] [INFO] [1749468348.261749490] [asr]: 0
[asr-5] [INFO] [1749468348.331305847] [asr]: 0
[asr-5] [INFO] [1749468348.420669705] [asr]: 0
[asr-5] [INFO] [1749468348.511797468] [asr]: 0
[asr-5] [INFO] [1749468348.601264784] [asr]: 0
[asr-5] [INFO] [1749468348.690582945] [asr]: 0
[asr-5] [INFO] [1749468348.781784501] [asr]: 0
[asr-5] [INFO] [1749468348.871133447] [asr]: 0
[asr-5] [INFO] [1749468348.962600416] [asr]: 0
[asr-5] [INFO] [1749468349.059499934] [asr]: 0
[asr-5] [INFO] [1749468349.151681477] [asr]: 0
[asr-5] [INFO] [1749468351.500270975] [asr]: 请你帮我把面前的红色方块放在蓝色方块的右边。
[asr-5] [INFO] [1749468351.501745499] [asr]: okay😊, let me think for a moment...
[model_service-3] [INFO] [1749468357.419215165] [model_service]: The upcoming task to be carried out : 1.
    调用视觉函数获取当前画面
[model_service-3] 2. 识别并定位红色方块和蓝色方块的位置
[model_service-3] 3. 调用机械臂夹取红色方块
[model_service-3] 4. 向右平移一段距离（10厘米）
[model_service-3] 5. 调用机械臂放下红色方块
[model_service-3] [INFO] [1749468359.374614593] [model_service]: "action": ['seewhat()'], "response": 好的，我先来看看面前都有些什么。
[model_service-3] [INFO] [1749468370.340389629] [model_service]: "action": ['grasp_obj(280,305,330,365)'],
    "response": 我已经找到了红色方块和蓝色方块，现在准备夹取红色方块。
[kin_srv-2] [WARN] [1749468379.684991427] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[action_service-4] start
[action_service-4] [INFO] [1749468382.813927217] [image converter]: Received: [280, 305, 330, 365]

```

When the robot completes the task, it enters the waiting state. Wake up the robot again and issue the command "End current task" to let the robot end the current task.

```

jetson@yahboom: ~
ra dummy link to your URDF.
[kin_srv-2] [WARN] [1749468392.263291779] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[kin_srv-2] [WARN] [1749468392.270509420] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[kin_srv-2] [WARN] [1749468392.274849310] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[kin_srv-2] [WARN] [1749468392.278826569] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[kin_srv-2] [WARN] [1749468392.282646994] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[kin_srv-2] [WARN] [1749468392.286499835] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[kin_srv-2] [WARN] [1749468392.291867776] [kdl_parser]: The root link base_link has an inertia specified
    in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
    ra dummy link to your URDF.
[model_service-3] [INFO] [1749468404.122745981] [model_service]: "action": ['set_cmdvel(0, -0.1, 0, 1)']
    , "response": 红色方块已经在我的手中了，现在我将向右平移一段距离。
[model_service-3] [INFO] [1749468416.607734718] [model_service]: "action": ['putdown()'], "response": 我
    已经向右平移了，现在将红色方块放在蓝色方块的右边。
[model_service-3] [INFO] [1749468436.562131746] [model_service]: "action": ['finishtask()'], "response":
    红色方块已经成功放在蓝色方块的右边啦！任务完成，还有什么需要我帮忙的吗？

```

### 3.2.3 Case 3: "Please help me remove the machine code in front of you that is higher than 5 cm"

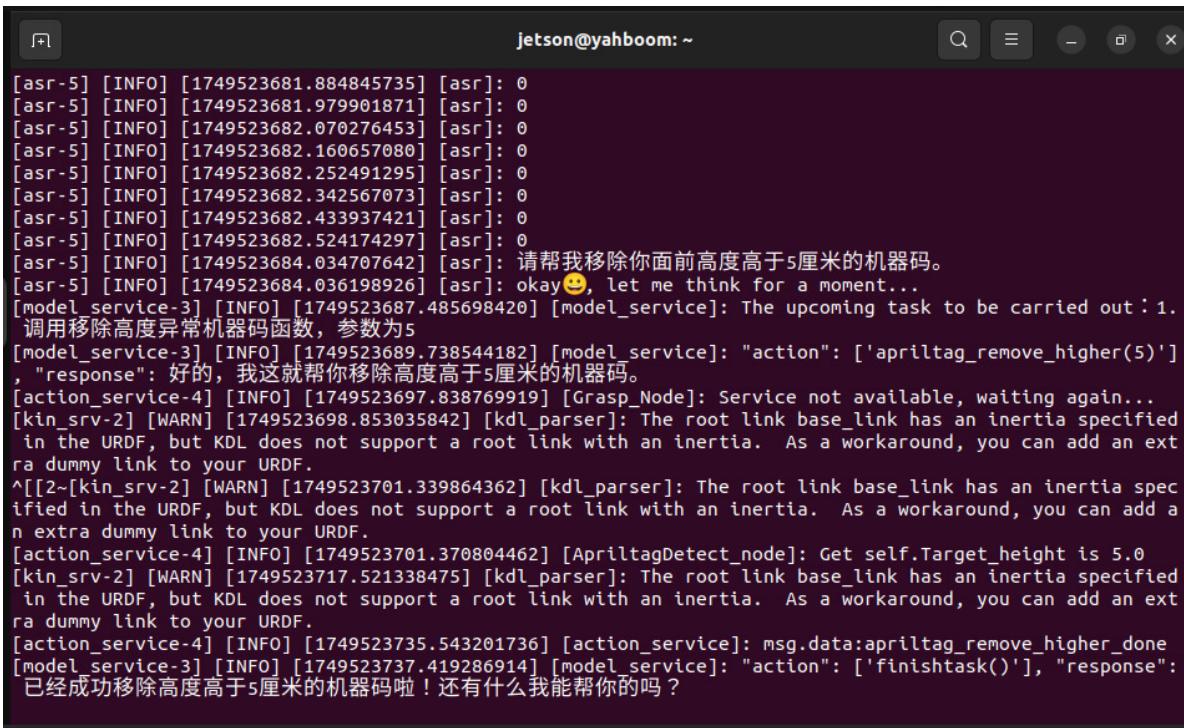
After waking up the robot, say the command "Please remove the machine code in front of you that is higher than 5 cm." The terminal prints the following information

```
[asr-5] [INFO] [1749523680.976392282] [asr]: 0
[asr-5] [INFO] [1749523681.068028257] [asr]: 0
[asr-5] [INFO] [1749523681.158435968] [asr]: 0
[asr-5] [INFO] [1749523681.249711888] [asr]: 0
[asr-5] [INFO] [1749523681.340122542] [asr]: 0
[asr-5] [INFO] [1749523681.430556686] [asr]: 0
[asr-5] [INFO] [1749523681.520831173] [asr]: 0
[asr-5] [INFO] [1749523681.611163328] [asr]: 0
[asr-5] [INFO] [1749523681.702387373] [asr]: 0
[asr-5] [INFO] [1749523681.793570903] [asr]: 0
[asr-5] [INFO] [1749523681.884845735] [asr]: 0
[asr-5] [INFO] [1749523681.979901871] [asr]: 0
[asr-5] [INFO] [1749523682.070276453] [asr]: 0
[asr-5] [INFO] [1749523682.160657080] [asr]: 0
[asr-5] [INFO] [1749523682.252491295] [asr]: 0
[asr-5] [INFO] [1749523682.342567073] [asr]: 0
[asr-5] [INFO] [1749523682.433937421] [asr]: 0
[asr-5] [INFO] [1749523682.524174297] [asr]: 0
[asr-5] [INFO] [1749523684.034707642] [asr]: 请帮我移除你面前高度高于5厘米的机器码。
[asr-5] [INFO] [1749523684.036198926] [asr]: okay😊, let me think for a moment...
[model_service-3] [INFO] [1749523687.485698420] [model_service]: The upcoming task to be carried out: 1. 调用移除高度异常机器码函数, 参数为5
[model_service-3] [INFO] [1749523689.738544182] [model_service]: "action": ['apriltag_remove_higher(5)']
, "response": 好的, 我这就帮你移除高度高于5厘米的机器码。
[action_service-4] [INFO] [1749523697.838769919] [Grasp_Node]: Service not available, waiting again...
[kin_srv-2] [WARN] [1749523698.853035842] [kdl_parser]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext ra dummy link to your URDF.
^[[2~
```

The terminal will open a window titled **result\_image**, where you can see the height of each machine code. After the distance measurement is stable, the robot will automatically adjust the distance between itself and the target, then use the robotic arm to grab the machine code at the target height and transfer it to the right side of the robot.



After the gripping is completed, the robot will enter the waiting state. Wake up the robot again and issue the command "end current task" to let the robot end the current task.



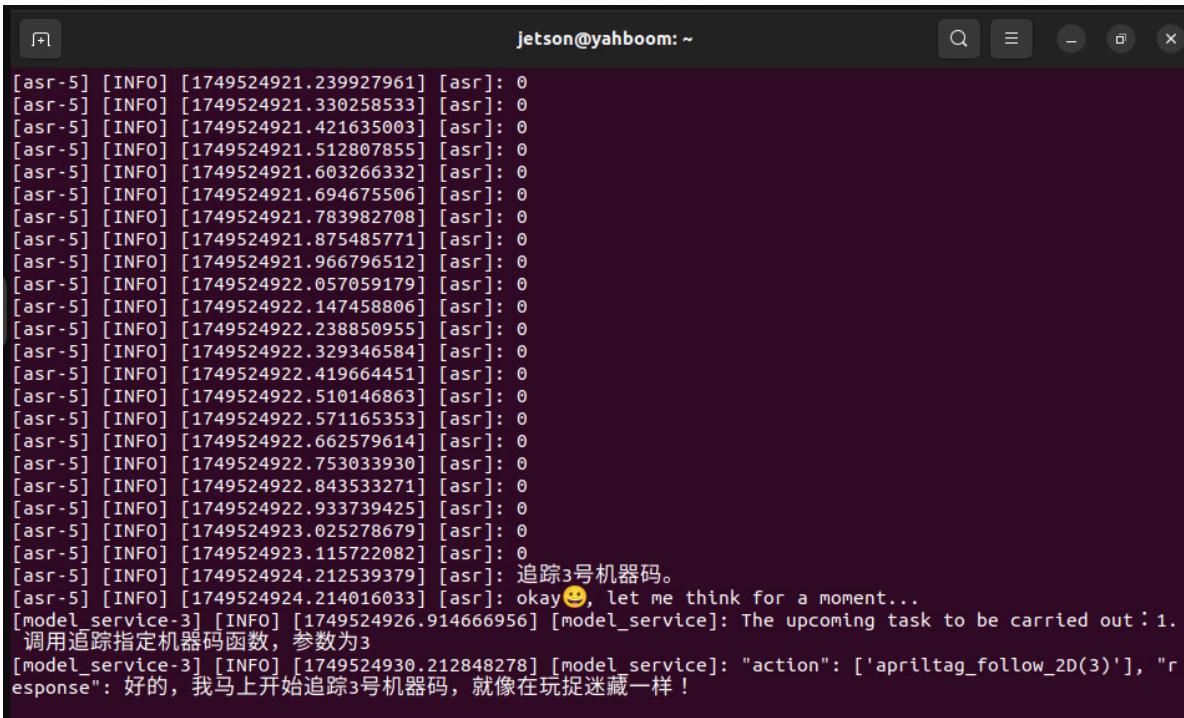
```

[asr-5] [INFO] [1749523681.884845735] [asr]: 0
[asr-5] [INFO] [1749523681.979901871] [asr]: 0
[asr-5] [INFO] [1749523682.070276453] [asr]: 0
[asr-5] [INFO] [1749523682.160657080] [asr]: 0
[asr-5] [INFO] [1749523682.252491295] [asr]: 0
[asr-5] [INFO] [1749523682.342567073] [asr]: 0
[asr-5] [INFO] [1749523682.433937421] [asr]: 0
[asr-5] [INFO] [1749523682.524174297] [asr]: 0
[asr-5] [INFO] [1749523684.034707642] [asr]: 请帮我移除你面前高度高于5厘米的机器码。
[asr-5] [INFO] [1749523684.036198926] [asr]: okay😊, let me think for a moment...
[model_service-3] [INFO] [1749523687.485698420] [model_service]: The upcoming task to be carried out: 1.
    调用移除高度异常机器码函数，参数为5
[model_service-3] [INFO] [1749523689.738544182] [model_service]: "action": ["apriltag_remove_higher(5)"]
, "response": 好的，我就帮你移除高度高于5厘米的机器码。
[action_service-4] [INFO] [1749523697.838769919] [Grasp_Node]: Service not available, waiting again...
[kin_srv-2] [WARN] [1749523698.853035842] [kdl_parser]: The root link base_link has an inertia specified
in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
ra dummy link to your URDF.
^[[2~[kin_srv-2] [WARN] [1749523701.339864362] [kdl_parser]: The root link base_link has an inertia spec
ified in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add a
n extra dummy link to your URDF.
[action_service-4] [INFO] [1749523701.370804462] [ApriltagDetect_node]: Get self.Target_height is 5.0
[kin_srv-2] [WARN] [1749523717.521338475] [kdl_parser]: The root link base_link has an inertia specified
in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an ext
ra dummy link to your URDF.
[action_service-4] [INFO] [1749523735.543201736] [action_service]: msg.data:apriltag_remove_higher_done
[model_service-3] [INFO] [1749523737.419286914] [model_service]: "action": ["finishtask()"], "response": 已经成功移除高度高于5厘米的机器码啦！还有什么我能帮你的吗？

```

### 3.2.4 Case 4: "Tracking Machine Number 3"

After waking up the robot, say the command "track machine code number 3" and the terminal will print the following information.

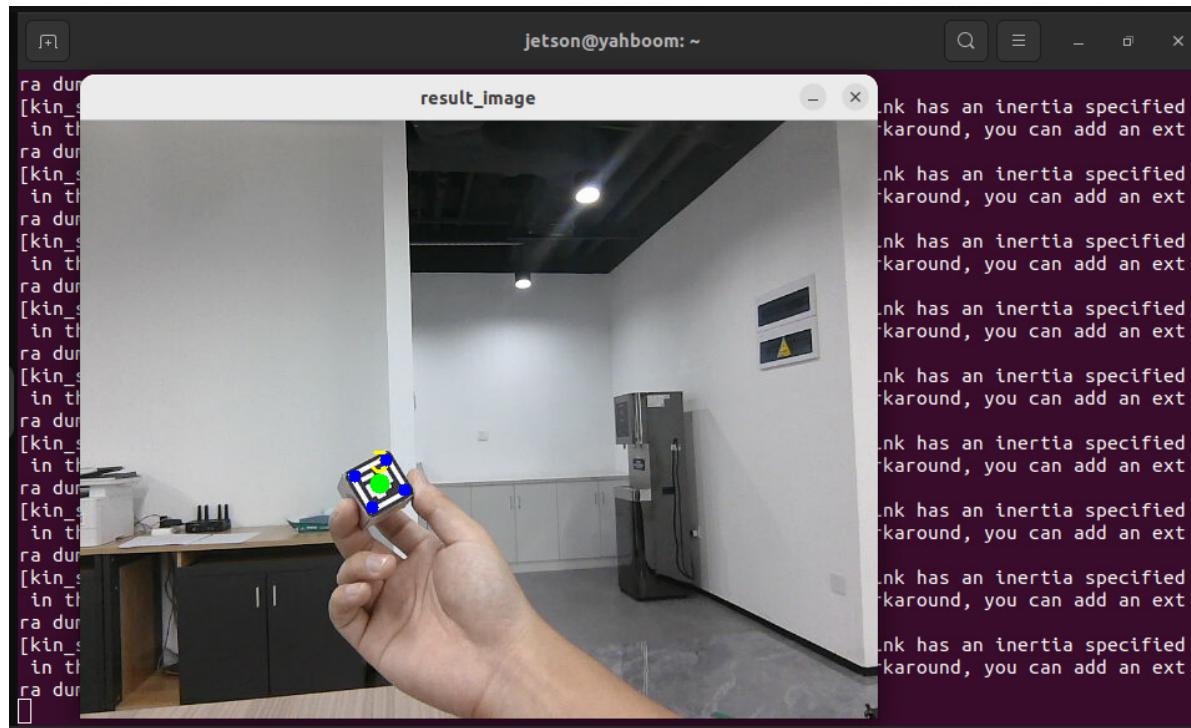


```

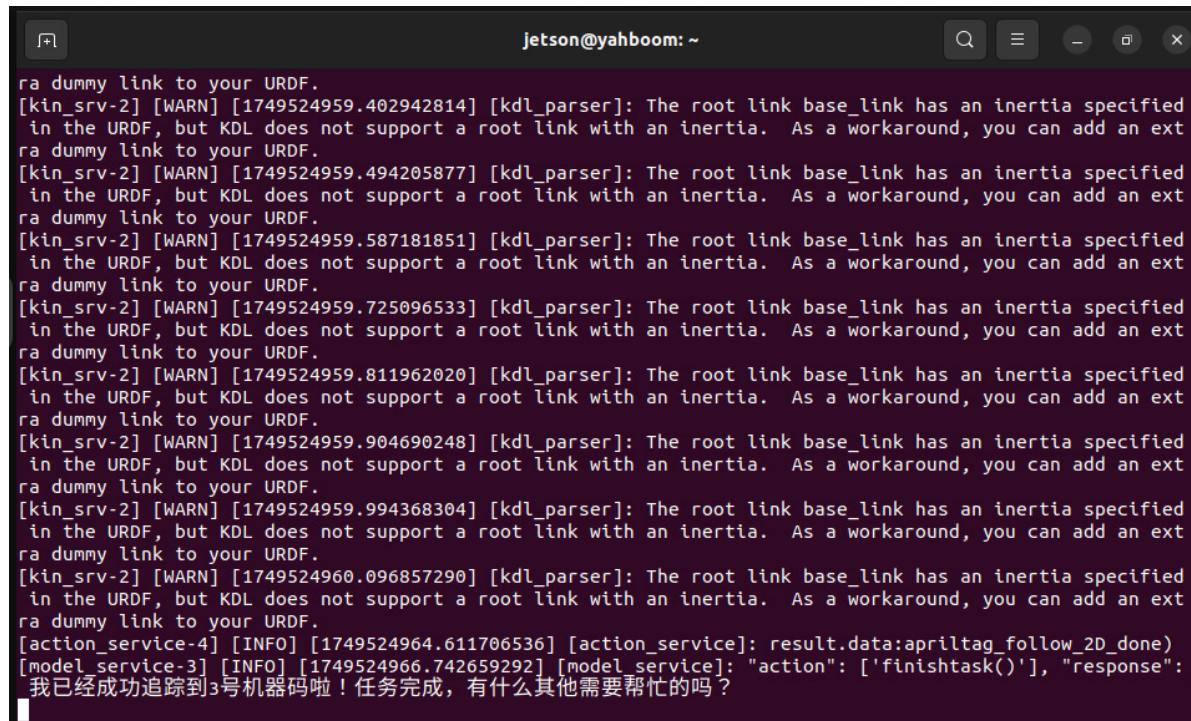
[asr-5] [INFO] [1749524921.239927961] [asr]: 0
[asr-5] [INFO] [1749524921.330258533] [asr]: 0
[asr-5] [INFO] [1749524921.421635003] [asr]: 0
[asr-5] [INFO] [1749524921.512807855] [asr]: 0
[asr-5] [INFO] [1749524921.603266332] [asr]: 0
[asr-5] [INFO] [1749524921.694675506] [asr]: 0
[asr-5] [INFO] [1749524921.783982708] [asr]: 0
[asr-5] [INFO] [1749524921.875485771] [asr]: 0
[asr-5] [INFO] [1749524921.966796512] [asr]: 0
[asr-5] [INFO] [1749524922.057059179] [asr]: 0
[asr-5] [INFO] [1749524922.147458806] [asr]: 0
[asr-5] [INFO] [1749524922.238850955] [asr]: 0
[asr-5] [INFO] [1749524922.329346584] [asr]: 0
[asr-5] [INFO] [1749524922.419664451] [asr]: 0
[asr-5] [INFO] [1749524922.510146863] [asr]: 0
[asr-5] [INFO] [1749524922.571165353] [asr]: 0
[asr-5] [INFO] [1749524922.662579614] [asr]: 0
[asr-5] [INFO] [1749524922.753033930] [asr]: 0
[asr-5] [INFO] [1749524922.843533271] [asr]: 0
[asr-5] [INFO] [1749524922.933739425] [asr]: 0
[asr-5] [INFO] [1749524923.025278679] [asr]: 0
[asr-5] [INFO] [1749524923.115722082] [asr]: 0
[asr-5] [INFO] [1749524924.212539379] [asr]: 追踪3号机器码。
[asr-5] [INFO] [1749524924.214016033] [asr]: okay😊, let me think for a moment...
[model_service-3] [INFO] [1749524926.914666956] [model_service]: The upcoming task to be carried out: 1.
    调用追踪指定机器码函数，参数为3
[model_service-3] [INFO] [1749524930.212848278] [model_service]: "action": ["apriltag_follow_2D(3)"], "re
sponse": 好的，我马上开始追踪3号机器码，就像在玩捉迷藏一样！

```

A window titled "result\_image" will open in the terminal . The robot's view will then follow the movement of the machine code. After the machine code stops moving, the robot will adjust its distance to the machine code and then grab the machine code in mid-air.



After the gripping is completed, the robot will give a voice prompt that the task is completed, the robotic arm is in the gripping state, and the robot enters the waiting state



Wake up the robot again and give it a rest command, and the robot will end its current task.

```

[asr-5] [INFO] [1749525442.261710362] [asr]: 0
[asr-5] [INFO] [1749525442.351883514] [asr]: 0
[asr-5] [INFO] [1749525442.442142842] [asr]: 0
[asr-5] [INFO] [1749525442.533290816] [asr]: 0
[asr-5] [INFO] [1749525442.624587751] [asr]: 0
[asr-5] [INFO] [1749525442.715874318] [asr]: 0
[asr-5] [INFO] [1749525442.786114287] [asr]: 0
[asr-5] [INFO] [1749525442.875165191] [asr]: 0
[asr-5] [INFO] [1749525442.965340007] [asr]: 0
[asr-5] [INFO] [1749525443.055741416] [asr]: 0
[asr-5] [INFO] [1749525443.145027458] [asr]: 0
[asr-5] [INFO] [1749525443.246293065] [asr]: 0
[asr-5] [INFO] [1749525443.337482127] [asr]: 0
[asr-5] [INFO] [1749525443.427827727] [asr]: 0
[asr-5] [INFO] [1749525443.519178070] [asr]: 0
[asr-5] [INFO] [1749525443.611979751] [asr]: 0
[asr-5] [INFO] [1749525443.702055334] [asr]: 0
[asr-5] [INFO] [1749525443.792618280] [asr]: 0
[asr-5] [INFO] [1749525443.886862721] [asr]: 0
[asr-5] [INFO] [1749525443.975249717] [asr]: 0
[asr-5] [INFO] [1749525444.065299476] [asr]: 0
[asr-5] [INFO] [1749525444.156242264] [asr]: 0
[asr-5] [INFO] [1749525444.247666879] [asr]: 0
[asr-5] [INFO] [1749525444.337344763] [asr]: 0
[asr-5] [INFO] [1749525445.552647106] [asr]: 你先去休息吧。
[asr-5] [INFO] [1749525445.554666575] [asr]: okay😊, let me think for a moment...
[model_service-3] [INFO] [1749525447.356658443] [model_service]: "action": ["finish_dialogue()"], "response": 好的，那我先去休息一会儿。有需要的时候再叫我哦！

```

## 4. Source code analysis

The source code is located at:

jetson orin nano, jetson orin NX host:

```
/home/jetson/M3Pro_ws/src/largemode1/largemode1/action_service.py
```

Jetson Nano, Raspberry Pi host:

You need to enter docker first

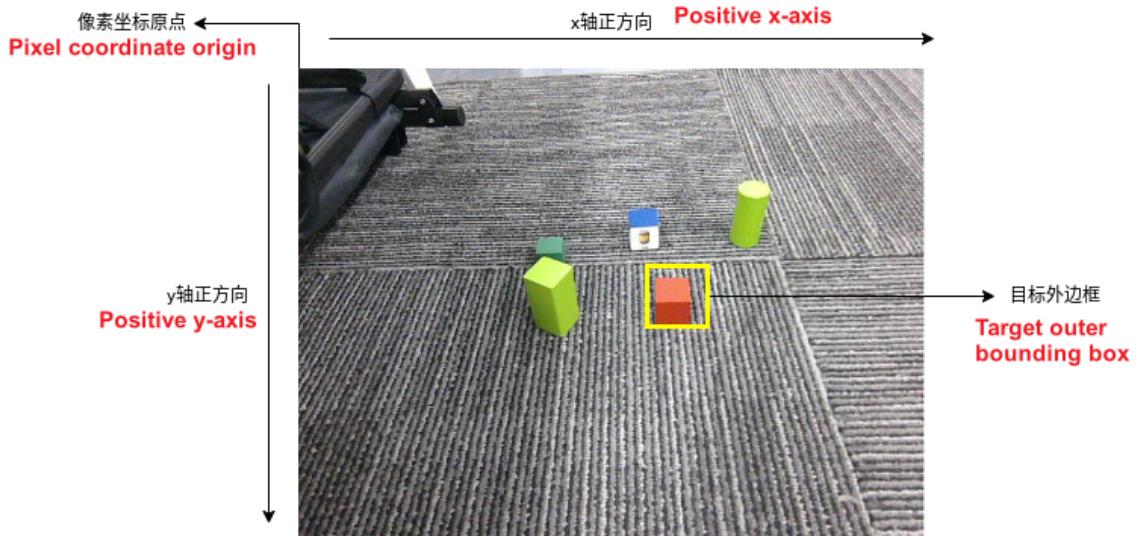
```
root/M3Pro_ws/src/largemode1/largemode1/action_service.py
```

### 4.1 Case 1

action\_service.py program:

**In Example 1, the seewhat and grasp\_obj methods in the CustomActionServer class are used . The seewhat function mainly obtains the color image of the depth camera. It has been explained in the Multimodal Visual Understanding section. Here, the grasp\_obj function is explained.**

The coordinate rules of objects in the robot arm's grasping perspective are shown in the figure below:



In the init\_ros\_comunication method of **the CustomActionServer** class, a topic publisher named corner\_xy is created to publish the coordinates of the outer frame of the target to be gripped, and a topic subscriber named largemode1\_arm\_done is created to receive the signal that the robotic arm has completed gripping.

```
# Create an object position publisher and publish the coordinates of the object
# to be clamped
self.object_position_pub = self.create_publisher(Int16MultiArray, 'corner_xy', 1)
#Create a robot arm to complete the topic subscriber
self.largemode1_arm_done_sub =
    self.create_subscription(String, '/largemode1_arm_done', self.largemode1_arm_
callback, 1)
```

The **grasp\_obj(x1, y1, x2, y2)** function is used to call the robotic arm to grasp the target object. The parameters are the coordinates of the upper left vertex and lower right vertex of the outer border of the object to be grasped (the upper left corner of the image is the pixel coordinate origin). For example, the coordinates of the outer border of the red square to be grasped in Case 1 can be known from the response of the large model: the coordinates of the upper left corner are (365,200) and the coordinates of the lower right corner are (408,261).

grasp\_obj starts three child processes: **grasp\_desktop**, **KCF\_follow**, and **ALM\_KCF\_Tracker\_Node**, and passes the parameters given by the AI model to the **ALM\_KCF\_Tracker\_Node** node through the topic. It then `rclpy.spin_until_future_complete(self, self.grasp_obj_future)` blocks the main process and waits for the grasping to complete.

```
def grasp_obj ( self , x1 , y1 , x2 , y2 ):
    process_1 = subprocess . Popen ([ 'ros2' , 'run' , 'largemode1_arm' ,
    'grasp_desktop' ])
    time.sleep ( 2.0 ) #Sleep for 2 seconds to wait for the thread to stabilize
```

```

        process_2 = subprocess . Popen ([ 'ros2' , 'run' , 'largemode1_arm' ,
'KCF_follow' ])
        process_3 = subprocess . Popen ([ 'ros2' , 'run' , 'M3Pro_KCF' ,
'ALM_KCF_Tracker_Node' ])
        time . sleep ( 3.0 ) #Sleep for 3 seconds to wait for the thread to
stabilize
        # Convert the argument from a string to an integer
        x1 = int ( x1 )
        y1 = int ( y1 )
        x2 = int ( x2 )
        y2 = int ( y2 )
        msg1 = Int16MultiArray ( data = [ x1 , y1 , x2 , y2 ] )
        self . bject_position_pub . publish ( msg1 )
        rclpy . spin_until_future_complete ( self , self . grasp_obj_future )
        result = self . grasp_obj_future . result ()
        if result . data == 'grasp_obj_done' :
            self . action_status_pub ( f'Robot feedback: Execution of
grasp_obj({x1},{y1},{x2},{y2}) completed' )
        else :
            self . action_status_pub ( f'Robot feedback: Execution of
grasp_obj({x1},{y1},{x2},{y2}) failed' )
            self . kill_process_tree ( process_1 . pid )
            self . kill_process_tree ( process_2 . pid )
            self . kill_process_tree ( process_3 . pid )
        self . grasp_obj_future = Future () # Reset the Future object

```

After the grasp is completed, **the KCF\_follow** node will publish a signal of **the largemode1\_arm\_done** topic with the content " **grasp\_obj\_done** ", set the **grasp\_obj\_future** object in the **largemode1\_arm\_done\_callback** callback function , and then exit the blocking in the **grasp\_obj** function. Then, the **kill\_process\_tree** method is called to recursively kill the process tree of the child process, and finally the status of the execution action is fed back to the execution layer large model. `rclpy.spin_until_future_complete(self, self.grasp_obj_future)`

```

def largemode1_arm_done_callback ( self , msg ): #Callback function for the
robotic arm to complete the grabbing process
    if msg . data in [ 'apriltag_sort_done' , ' apriltag_sort_failed' ]:
        if not self . apriltag_sort_future . done ():
            self . apriltag_sort_future . set_result ( msg )
    elif msg . data == 'apriltag_follow_2D_done' :
        if not self . apriltag_follow_2D_future . done ():
            self . apriltag_follow_2D_future . set_result ( msg )
    elif msg . data in [ 'apriltag_remove_higher_done' ,
'apriltag_remove_higher_failed' ]:
        self . get_logger () . info ( f'msg.data:{msg.data}' )
        if not self . apriltag_remove_higher_future . done ():
            self . apriltag_remove_higher_future . set_result ( msg )
    elif msg . data == 'color_follow_2D_done' :
        if not self . color_follow_2D_future . done ():
            self . color_follow_2D_future . set_result ( msg )
    elif msg . data == 'color_sort_done' :
        if not self . color_sort_future . done ():
            self . color_sort_future . set_result ( msg )
    elif msg . data == 'grasp_obj_done' :

```

```

        if not self . grasp_obj_future . done () :
            self . grasp_obj_future . set_result ( msg )
        elif msg . data == 'color_remove_higher_done' :
            if not self . color_remove_higher_future . done () :
                self . color_remove_higher_future . set_result ( msg )
        elif msg . data == 'follow_line_clear_future_done' :
            if not self . follow_line_clear_future . done () :
                self . follow_line_clear_future . set_result ( msg )

```

## 4.2 Case 2

action\_service.py program:

Case 2 uses **the seewhat**, **grab\_obj**, and **set\_cmdvel** methods in **the CustomActionServer** class. **The seewhat** and **grab\_obj** functions have been explained in Case 1. Here we explain the newly appeared **set\_cmdvel** function.

**set\_cmdvel** is a function that controls the movement of the robot chassis. It does this by setting the x- and y-axis speeds and the z-axis angular velocity, publishing the cmd\_vel topic, and then feeding back the status of the execution action to the execution layer model.

```

#Create a speed topic publisher
self . publisher = self . create_publisher ( Twist , self . Speed_topic , 10 )

def set_cmdvel ( self , linear_x , linear_y , angular_z , duration ): #Publish
cmd_vel
    # Convert the argument from string to floating point
    linear_x = float ( linear_x )
    linear_y = float ( linear_y )
    angular_z = float ( angular_z )
    duration = float ( duration )
    twist = Twist ()
    twist . linear . x = linear_x
    twist . linear . y = linear_y
    twist . angular . z = angular_z
    self . _execute_action ( twist , duration )
    self . stop ()
    if self . combination_mode : #Is it combination mode? The combination mode
needs to be executed.
        return
    else :
        self . action_status_pub ( f'Robot feedback: Execution of
set_cmdvel({linear_x},{linear_y},{angular_z},{duration}) completed' )

```

## 4.3 Case 3

action\_service.py program:

Case 3, using the **apriltag\_remove\_higher** method in **the CustomActionServer** class

The program principle is similar to that of Case 1. The child process is started as **the grab\_desktop\_remove** and **apriltag\_remove\_higher** nodes, and is blocked until the **largemodel\_arm\_done\_callback** callback function receives " **apriltag\_remove\_higher\_done** " or " **apriltag\_remove\_higher\_failed** ". Then, it exits the blocking, kills the process tree of the child process, and then feeds back the status of the execution action to the execution layer large model.

```
def apriltag_remove_higher(self,target_high):#Remove the machine code of the specified height
    target_highf = float(target_high)/100
    process_1 = subprocess.Popen(['ros2', 'run', 'largemodel_arm',
    'grasp_desktop_remove'])
    time.sleep(4)#wait for thread to stabilize
    process_2 = subprocess.Popen(['ros2', 'run', 'largemodel_arm',
    'apriltag_remove_higher','--ros-args',' -p',f'target_high:={target_highf:.2f}'])
    rclpy.spin_until_future_complete(self, self.apriltag_remove_higher_future)
    result=self.apriltag_remove_higher_future.result()
    if result.data== 'apriltag_remove_higher_done':
        self.action_status_pub(f'机器人反馈: 执行
apriltag_remove_higher({target_high})完成')
    elif result.data== 'apriltag_remove_higher_failed':
        self.action_status_pub(f'机器人反馈: 执行执行
apriltag_remove_higher({target_high})失败')
        self.kill_process_tree(process_1.pid)
        self.kill_process_tree(process_2.pid)
        self.apriltag_remove_higher_future = Future() # Reset the Future object
        self.pubsix_Arm(self.init_joints)
```

## 4.4 Case 4

action\_service.py program:

Case 4, using the **apriltag\_follow\_2D** method in **the CustomActionServer class**

The program principle is similar to the previous cases. The subprocesses started are **grab** and **apriltag\_follow\_2D**. The subprocesses will block the program until **the largemodel\_arm\_done\_callback** callback function receives " **apriltag\_follow\_2D\_done** ", and then feedback the status of the execution action to the execution layer large model.

```
def apriltag_follow_2D(self,target_id):#Tracking machine code
    target_idf = float(target_id)
    process_1 = subprocess.Popen(['ros2', 'run', 'largemodel_arm', 'grasp'])
    process_2 = subprocess.Popen(['ros2', 'run', 'largemodel_arm',
    'apriltag_follow_2D','--ros-args',' -p',f'target_id:={target_idf:.1f}'])
    rclpy.spin_until_future_complete(self, self.apriltag_follow_2D_future)
    result=self.apriltag_follow_2D_future.result()
    self.get_logger().info(f'result.data:{result.data}')
    if result.data =='apriltag_follow_2D_done':
        self.action_status_pub(f'机器人反馈:执行apriltag_follow_2D({target_id})完
成')
    else:
        self.action_status_pub(f'机器人反馈:执行执行apriltag_follow_2D({target_id})失
败')
    self.kill_process_tree(process_1.pid)
```

```
self.kill_process_tree(process_2.pid)
self.apriltag_follow_2D_future = Future() # Reset the Future object
```