

# Multimodal visual understand

## 1. Course Content

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

## 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.

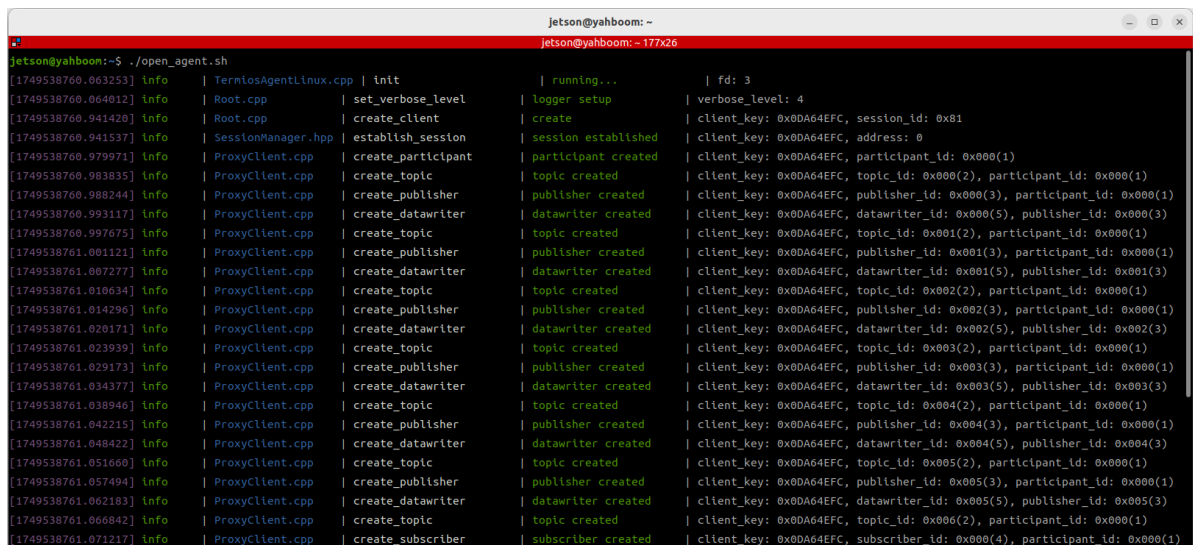
### 2.2 Start the Agent

**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 | logger setup | verbose_level: 4  
[1749538760.941420] Info | Root.cpp | create_client | create | client_key: 0x0DA64EFC, session_id: 0x81  
[1749538760.941537] Info | SessionManager.hpp | establish_session | session established | client_key: 0x0DA64EFC, address: 0  
[1749538760.979971] Info | ProxyClient.cpp | create_participant | participant created | client_key: 0x0DA64EFC, participant_id: 0x000(1)  
[1749538760.983835] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x001(2), participant_id: 0x000(1)  
[1749538760.988244] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x000(3), participant_id: 0x000(1)  
[1749538760.993117] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x000(5), publisher_id: 0x000(3)  
[1749538760.997675] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x001(2), participant_id: 0x000(1)  
[1749538761.001121] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x001(3), participant_id: 0x000(1)  
[1749538761.007277] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x001(5), publisher_id: 0x001(3)  
[1749538761.010634] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x002(2), participant_id: 0x000(1)  
[1749538761.014296] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x002(3), participant_id: 0x000(1)  
[1749538761.020171] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x002(5), publisher_id: 0x002(3)  
[1749538761.023939] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x003(2), participant_id: 0x000(1)  
[1749538761.029173] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x003(3), participant_id: 0x000(1)  
[1749538761.034377] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x003(5), publisher_id: 0x003(3)  
[1749538761.038946] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x004(2), participant_id: 0x000(1)  
[1749538761.042215] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x004(3), participant_id: 0x000(1)  
[1749538761.048422] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x004(5), publisher_id: 0x004(3)  
[1749538761.051660] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x005(2), participant_id: 0x000(1)  
[1749538761.057494] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x0DA64EFC, publisher_id: 0x005(3), participant_id: 0x000(1)  
[1749538761.062183] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x0DA64EFC, datawriter_id: 0x005(5), publisher_id: 0x005(3)  
[1749538761.066842] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x0DA64EFC, topic_id: 0x006(2), participant_id: 0x000(1)  
[1749538761.071217] Info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x0DA64EFC, subscriber_id: 0x000(4), participant_id: 0x000(1)
```

## 3. Run the case

## 3.1 Startup Program

Open the terminal on the vehicle and enter the command:

```
ros2 launch largemodel largemodel_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 :paraformer-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 reference test cases, users can write their own dialogue instructions

- Please show me what items are in front of you and tell me what they are used for.
- Please look in front of you to see if there is a blue cube and a pack of tissues. If yes, nod your head. If not, shake your head.

### 3.2.1 Case 1

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

```
[asr-5] [INFO] [1755161550.508637641] [asr]: -  
[asr-5] [INFO] [1755161550.568579598] [asr]: -  
[asr-5] [INFO] [1755161550.630618133] [asr]: -  
[asr-5] [INFO] [1755161550.689368330] [asr]: -  
[asr-5] [INFO] [1755161550.751563392] [asr]: -  
[asr-5] [INFO] [1755161553.406620218] [asr]: Please show me what items are in front of you and tell me what they are used for.  
[asr-5] [INFO] [1755161553.408079937] [asr]: 😊Okay, let me think for a moment...  
[model_service-3] [INFO] [1755161557.409736123] [model_service]: Decision making AI planning:1. Obtain the image from the current pe  
rspective.  
[model_service-3] 2. Analyze the image to identify the items in front of the robot.  
[model_service-3] 3. Describe the items and their uses based on the analysis.  
[model_service-3] [INFO] [1755161559.801961593] [model_service]: "action": ['seewhat()'], "response": Okay, I'm going to take a look  
around and see what's in front of me. Just a second!  
[action_service-4] [ERROR] [1755161568.400131161] [action_service]: The image is being saved and no new information will be accepted  
[action_service-4] [ERROR] [1755161568.433579917] [action_service]: The image is being saved and no new information will be accepted  
[action_service-4] [ERROR] [1755161568.476371215] [action_service]: The image is being saved and no new information will be accepted  
[action_service-4] [ERROR] [1755161568.513242617] [action_service]: The image is being saved and no new information will be accepted  
[model_service-3] [INFO] [1755161572.624373349] [model_service]: "action": [], "response": I see a few items here! There's a pack of  
wet wipes, which are great for cleaning hands or surfaces quickly. Next to it is a water bottle, perfect for staying hydrated. And  
there's also a yellow thermos, probably for keeping drinks warm or cold. All very practical items for daily use!  
[action_service-4] [INFO] [1755161600.823410595] [action_service]: Published message: Robot feedback: Reply to user completed  
[model_service-3] [INFO] [1755161606.022631604] [model_service]: "action": ['finishtask()'], "response": That's all I've got to shar  
e for now! Let me know if you'd like to explore something else.
```

Robot's perspective:





The robot executes the `finishtask()` function, proving that it has completed the user's instructions and entered **the waiting state**. After the user wakes up the robot again, they can choose to end the current task cycle and start a new task, or continue testing in the current task cycle. In this example, case 2 is tested directly in a task cycle (note that the robot is required to "observe the environment again").

### 3.2.2 Case 2

Wake up the robot, after the buzzer sounds briefly (beep—) after answering, say the test command, the terminal prints the following information

```
[asr-5] [INFO] [1755162260.725397985] [asr]: -
[asr-5] [INFO] [1755162260.788760277] [asr]: -
[asr-5] [INFO] [1755162260.852043142] [asr]: -
[asr-5] [INFO] [1755162260.919857709] [asr]: -
[asr-5] [INFO] [1755162260.972477169] [asr]: -
[asr-5] [INFO] [1755162261.033573001] [asr]: -
[asr-5] [INFO] [1755162261.094712098] [asr]: -
[asr-5] [INFO] [1755162265.622950753] [asr]: Please look in front of you to see if there is a blue cube and a pack of tissues, if yes, nod your head, if not, shake your head.
[asr-5] [INFO] [1755162265.624465406] [asr]: 😊Okay, let me think for a moment...
[model_service-3] [INFO] [1755162270.202202367] [model_service]: "action": ['seewhat()'], "response": 'I'm checking the scene right now, looking for that blue cube and tissues. Let me see what's in front of me!'
[action_service-4] [ERROR] [1755162271.507106353] [action_service]: The image is being saved and no new information will be accepted
[action_service-4] [ERROR] [1755162271.513729337] [action_service]: The image is being saved and no new information will be accepted
[action_service-4] [ERROR] [1755162271.562527185] [action_service]: The image is being saved and no new information will be accepted
[model_service-3] [INFO] [1755162276.939325113] [model_service]: "action": ['arm_shake()'], "response": 'I see the tissues, but there's a red cube here, not a blue one. So no blue cube in sight-shaking my head to let you know!'
[action_service-4] [INFO] [1755162277.608206735] [action_service]: Published message: Robot feedback: Execution ['seewhat()'], 'arm_shake()'] completed
[model_service-3] [INFO] [1755162281.876884315] [model_service]: "action": ['finishtask()'], "response": 'I've checked and confirmed-no blue cube here, just the tissues and a red one. All done!'
```

Robot's perspective:



## 4. Source code analysis

---

The source code is located at:

jetson orin nano, jetson orin NX host:

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

```
/home/jetson/M3Pro_ws/src/largemodel/largemodel/model_service.py
```

Jetson Nano, Raspberry Pi host:

You need to enter docker first

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

```
root/M3Pro_ws/src/largemodel/largemodel/model_service.py
```

action\_service.py program:

In the `init_ros_communication` method of the `CustomActionServer` class, a topic publisher called `seewhat_handle` is created to publish visual processing signals.



```
# Create a publisher and publish the seewhat_handle topic
self.seewhat_handle_pub = self.create_publisher(String, 'seewhat_handle', 1)
```

When the action server calls the seewhat method, it publishes a signal to the seewhat\_handle topic, notifying the model server model\_service node to upload an image to the multimodal large model.

```
def seewhat(self):
    msg = String(data='seewhat')
    self.seewhat_handle_pub.publish(msg)#Normalize, publish the seewhat topic,
    and let the master node call the large model
```

model\_service.py program:

In the init\_ros\_comunication method of the LargeModelService class

Created a seewhat\_handle topic subscriber to receive visual processing signals

Created a camera/color/image\_raw topic subscriber to receive the color image from the depth camera

```
#Create a seewhat subscriber
self.seewhat_sub = self.create_subscription(String, 'seewhat_handle',
self.seewhat_callback,1)
#ImageTopicSubscribers
self.subscription = self.create_subscription(Image,self.image_topic,
self.image_callback,1)
```

When the seewhat\_callback callback function receives the visual processing signal, it calls the dual\_large\_model\_mode method to first save an image from the depth camera and then upload it to the multimodal large model.

```
def seewhat_callback(self,msg):
    if msg.data == "seewhat":
        if self.use_double_llm:#Online model inference method: decision layer
reasoning + execution layer supervision
            self.dual_large_model_mode(type="image")
        else:#single model reasoning mode
            self.single_largeModel_infer(image_path=self.image_save_path)
```

The image\_callback callback function will continue to subscribe to the color image of the depth camera. When the image needs to be saved, the save\_single\_image method will be called to save an image.png image file.

Image save path:

jetson orin nano, jetson orin NX host:

```
/home/jetson/M3Pro_ws/install/largemodel/share/largemodel/resources_file/image.p
ng
```

Jetson Nano and Raspberry Pi host need to enter docker first:

```
root/M3Pro_ws/install/largemodel/share/largemodel/resources_file/image.png
```

```
def image_callback(self, msg):#Image callback function
    self.image_msg = msg

def save_single_image(self):
    if self.image_msg is None:
        self.get_logger().warning('No image received yet.')
        return
    try:
        #Image callback function
        cv_image = self.bridge.imgmsg_to_cv2(self.image_msg, 'bgr8')
        # Save the image
        cv2.imwrite(self.image_save_path, cv_image)
        # self.get_logger().info(f'Saved image: {self.image_save_path}')
    except Exception as e:
        self.get_logger().error(f'Error saving image: {e}')
```