Multimodal Large Model + Robotic Arm Color Block Sorting（Text Version）

Before running the function, you need to close the App and large programs. For the closing method, refer to [4. Preparation] - [1. Manage APP control services].

1. Function Description

After the program runs, input the color block sorting order through the terminal. The large model will plan the steps to complete the color block sorting task, then the program will control the robotic arm to sort the color blocks according to the order and place them at the set positions.

2. Startup

Users with Jetson-Nano board version need to enter the docker container and input the following command. Orin board users can directly open the terminal and input the following command:

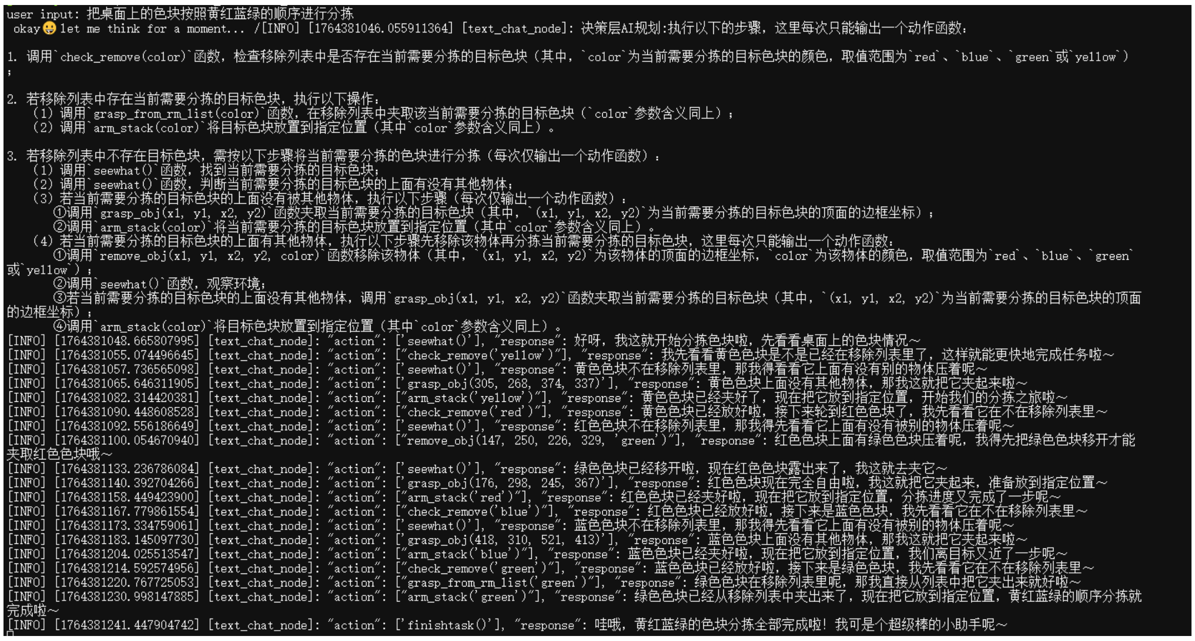
```
ros2 launch largemodel largemodel_control.launch.py text_chat_mode:=True
```

Then open a second terminal and input the following command:

```
ros2 run text_chat text_chat
```

Then specify the color block sorting order in the terminal, you can refer to the following example:

```
Sort the color blocks on the desktop in the order of yellow, red, blue, green
```



```
user input: 把桌面上的色块按照黄红蓝绿的顺序进行分拣
 okay😛let me think for a moment... /[INFO] [1764381046.055911364] [text_chat_node]: 决策层AI规划:执行以下的步骤，这里每次只能输出一个动作函数：

1. 调用`check_remove(color)`函数，检查移除列表中是否存在当前需要分拣的目标色块（其中，`color`为当前需要分拣的目标色块的颜色，取值范围为`red`、`blue`、`green`或`yellow`）
；

2. 若移除列表中存在当前需要分拣的目标色块，执行以下操作：
   （1）调用`grasp_from_rm_list(color)`函数，在移除列表中夹取该当前需要分拣的目标色块（`color`参数含义同上）；
   （2）调用`arm_stack(color)`将目标色块放置到指定位置（其中`color`参数含义同上）。

3. 若移除列表中不存在目标色块，需按以下步骤将当前需要分拣的色块进行分拣（每次仅输出一个动作函数）：
   （1）调用`seewhat()`函数，找到当前需要分拣的目标色块；
   （2）调用`seewhat()`函数，判断当前需要分拣的目标色块的上面有没有其他物体；
   （3）若当前需要分拣的目标色块的上面没有被其他物体，执行以下步骤（每次仅输出一个动作函数）：
      ①调用`grasp_obj(x1, y1, x2, y2)`函数夹取当前需要分拣的目标色块（其中，`(x1, y1, x2, y2)`为当前需要分拣的目标色块的顶面的边框坐标）；
      ②调用`arm_stack(color)`将当前需要分拣的目标色块放置到指定位置（其中`color`参数含义同上）。
   （4）若当前需要分拣的目标色块的上面有其他物体，执行以下步骤先移除该物体再分拣当前需要分拣的目标色块，这里每次只能输出一个动作函数：
      ①调用`remove_obj(x1, y1, x2, y2, color)`函数移除该物体（其中，`(x1, y1, x2, y2)`为该物体的顶面的边框坐标，`color`为该物体的颜色，取值范围为`red`、`blue`、`green`
或`yellow`）；
      ②调用`seewhat()`函数，观察环境；
      ③若当前需要分拣的目标色块的上面没有其他物体，调用`grasp_obj(x1, y1, x2, y2)`函数夹取当前需要分拣的目标色块（其中，`(x1, y1, x2, y2)`为当前需要分拣的目标色块的顶面
的边框坐标）；
      ④调用`arm_stack(color)`将目标色块放置到指定位置（其中`color`参数含义同上）。
[INFO] [1764381048.665807995] [text_chat_node]: "action": ['seewhat()'], "response": 好呀，我这就开始分拣色块啦，先看看桌面上的色块情况~
[INFO] [1764381055.074496645] [text_chat_node]: "action": ["check_remove('yellow')"], "response": 我先看看黄色色块是不是已经在移除列表里了，这样就能更快地完成任务啦~
[INFO] [1764381057.736565098] [text_chat_node]: "action": ['seewhat()'], "response": 黄色色块不在移除列表里，那我得看看它上面有没有别的物体压着呢~
[INFO] [1764381065.646311905] [text_chat_node]: "action": ['grasp_obj(305, 268, 374, 337)'], "response": 黄色色块上面没有其他物体，那我这就把它夹起来啦~
[INFO] [1764381082.314420381] [text_chat_node]: "action": ["arm_stack('yellow')"], "response": 黄色色块已经夹好了，现在把它放到指定位置，开始我们的分拣之旅啦~
[INFO] [1764381090.448608528] [text_chat_node]: "action": ["check_remove('red')"], "response": 黄色色块已经放好啦，接下来轮到红色色块了，我先看看它在不在移除列表里~
[INFO] [1764381092.556186649] [text_chat_node]: "action": ['seewhat()'], "response": 红色色块不在移除列表里，那我得先看看它上面有没有被别的物体压着呢~
[INFO] [1764381100.054670940] [text_chat_node]: "action": ["remove_obj(147, 250, 226, 329, 'green')"], "response": 红色色块上面有绿色色块压着呢，我得先把绿色色块移开才能
夹取红色色块哦~
[INFO] [1764381133.236786084] [text_chat_node]: "action": ['seewhat()'], "response": 绿色色块已经移开啦，现在红色色块露出来了，我这就去夹它~
[INFO] [1764381140.392704266] [text_chat_node]: "action": ['grasp_obj(176, 298, 245, 367)'], "response": 红色色块现在完全自由啦，我这就把它夹起来，准备放到指定位置~
[INFO] [1764381158.449423900] [text_chat_node]: "action": ["arm_stack('red')"], "response": 红色色块已经夹好啦，现在把它放到指定位置，分拣进度又完成了一步呢~
[INFO] [1764381167.779861554] [text_chat_node]: "action": ["check_remove('blue')"], "response": 红色色块已经放好啦，接下来是蓝色色块，我先看看它在不在移除列表里~
[INFO] [1764381173.334759061] [text_chat_node]: "action": ['seewhat()'], "response": 蓝色色块不在移除列表里，那我得先看看它上面有没有被别的物体压着呢~
[INFO] [1764381183.145097730] [text_chat_node]: "action": ['grasp_obj(418, 310, 521, 413)'], "response": 蓝色色块上面没有其他物体，那我这就把它夹起来啦~
[INFO] [1764381204.025513547] [text_chat_node]: "action": ["arm_stack('blue')"], "response": 蓝色色块已经夹好啦，现在把它放到指定位置，我们离目标又近了一步呢~
[INFO] [1764381214.592574956] [text_chat_node]: "action": ["check_remove('green')"], "response": 蓝色色块已经放好啦，接下来是绿色色块，我先看看它在不在移除列表里~
[INFO] [1764381220.767725053] [text_chat_node]: "action": ["grasp_from_rm_list('green')"], "response": 绿色色块在移除列表里呢，那我直接从列表中把它夹出来就好啦~
[INFO] [1764381230.998147885] [text_chat_node]: "action": ["arm_stack('green')"], "response": 绿色色块已经从移除列表中夹出来了，现在把它放到指定位置，黄红蓝绿的顺序分拣就
完成啦~
[INFO] [1764381241.447904742] [text_chat_node]: "action": ['finishtask()'], "response": 哇哦，黄红蓝绿的色块分拣全部完成啦！我可是个超级棒的小助手呢~
```

As shown in the figure above, before sorting the yellow color block, first check if it is in the removal list. If it is, then directly grip it from the removal list and place it at the yellow color block placement point; if not, then determine if there are other color blocks on top of the yellow color block. If there are, first remove the other color blocks above the yellow color block. If there are none, then directly grip it and place it at the yellow color block placement point. Other color blocks are also sorted according to this execution order until the task is completed.

3. Task Planning

1. Call `check_remove(color)` function to check if the current target color block to be sorted exists in the removal list (where `color` is the color of the current target color block to be sorted, value range is `red`, `blue`, `green` or `yellow`);
2. If the current target color block to be sorted exists in the removal list, execute the following operations:
   (1) Call `grasp_from_rm_list(color)` function to grip the current target color block to be sorted from the removal list (meaning of `color` parameter is the same as above);
   (2) Call `arm_stack(color)` to place the target color block at the specified position (where `color` parameter meaning is the same as above).
3. If the target color block does not exist in the removal list, follow these steps to sort the current color block to be sorted (only output one action function at a time):
   (1) Call `seewhat()` function to find the current target color block to be sorted;
   (2) Call `seewhat()` function to determine if there are other objects on top of the current target color block to be sorted;
   (3) If there are no other objects on top of the current target color block to be sorted, execute the following steps (only output one action function at a time):
   ① Call `grasp_obj(x1, y1, x2, y2)` function to grip the current target color block to be sorted (where `(x1, y1, x2, y2)` are the top surface border coordinates of the current target color block to be sorted);
   ② Call `arm_stack(color)` to place the current target color block to be sorted at the specified position (where `color` parameter meaning is the same as above).
   (4) If there are other objects on top of the current target color block to be sorted, execute the following steps to first remove that object then sort the current target color block to be sorted, here only one action function can be output at a time:
   ① Call `remove_obj(x1, y1, x2, y2, color)` function to remove that object (where `(x1, y1, x2, y2)` are the top surface border coordinates of that object, `color` is the color of that object, value range is `red`, `blue`, `green` or `yellow`);
   ② Call `seewhat()` function to observe the environment;
   ③ If there are no other objects on top of the current target color block to be sorted, call `grasp_obj(x1, y1, x2, y2)` function to grip the current target color block to be sorted (where `(x1, y1, x2, y2)` are the top surface border coordinates of the current target color block to be sorted);
   ④ Call `arm_stack(color)` to place the target color block at the specified position (where `color` parameter meaning is the same as above).

4. Core Code Analysis

4.1. check_remove(color) Function

Source code path: `LargeModel_ws/src/largemodel/largemodel/action_service.py`

```
def check_remove(self,color):
    #Get current color parameter
    check_color = color.strip("'\"")
    #If the current color is in the self.cur_rm_pose dictionary, it means it's in
the removal list, otherwise, it was not found in the removal list
    if check_color in self.cur_rm_pose:
        self.action_status_pub("color is in the rm_list")
    else:
        self.action_status_pub("color is not in the rm_list")
#You can see what content color is in the rm_list and color is not in the rm_list
return to the large model respectively
"color is in the rm_list" : "Color block exists in removal list, grip this color
block from removal list",
"color is not in the rm_list" : "Color block does not exist in removal list,
determine if there are other objects on top of this color block"
```

4.2. grasp_from_rm_list Function

Source code path: `LargeModel_ws/src/largemodel/largemodel/action_service.py`

```
def grasp_from_rm_list(self,color):
    #Get current color parameter
    tar_color = color.strip("'\"")
    #Find corresponding key value in dictionary based on current color parameter,
assign to tar_joints
    tar_joints = self.cur_rm_pose.get(tar_color)
    #Control robotic arm six servos
    Arm.Arm_serial_servo_write6(tar_joints[0], tar_joints[1], tar_joints[2],
tar_joints[3], tar_joints[4], tar_joints[5],2000)
    time.sleep(2.0)
    Arm.Arm_serial_servo_write(6, 140, 1000)
    time.sleep(2.0)
    Arm.Arm_serial_servo_write6(90,120,10,10,90,140,2000)
    time.sleep(2.0)
    #Feedback to large model that gripping color block from removal list is
complete
    self.action_status_pub("grasp_from_rm_list_done")
```

4.3. arm_stack Function

Source code path: `LargeModel_ws/src/largemodel/largemodel/action_service.py`

```
def arm_stack(self,color):
    #Get current color parameter
    tar_color = color.strip("'\"")
    #Select robotic arm placement posture based on current color parameter
    if tar_color == 'red':
        Arm.Arm_serial_servo_write6(117, 19, 66, 56, 90,self.grasp_joint,2000)
        time.sleep(2.0)
    elif tar_color == 'green':
        Arm.Arm_serial_servo_write6(136, 66, 20, 29, 90,self.grasp_joint,2000)
        time.sleep(2.0)
    elif tar_color == 'blue':
        Arm.Arm_serial_servo_write6(44, 66, 20, 28, 90,self.grasp_joint,2000)
        time.sleep(2.0)
    elif tar_color == 'yellow':
        Arm.Arm_serial_servo_write6(65, 22, 64, 56, 90,self.grasp_joint,2000)
        time.sleep(2.0)
    Arm.Arm_serial_servo_write(6, 30, 1500)
    time.sleep(1.5)
    Arm.Arm_serial_servo_write6(90,120,10,10,90,30,2000)
    time.sleep(2.0)
    #Feedback to large model that color block placement is complete
    self.action_status_pub("arm_stack_done", color=color)
```

4.4. remove_obj Function

Source code path: `LargeModel_ws/src/largemodel/largemodel/action_service.py`

```
def remove_obj(self, x1, y1, x2, y2,color):
    #Get current color parameter
    cur_rm_name = color.strip("'\"")  # Remove single and double quotes
    self.cur_rm_name = cur_rm_name
    #Create a key in the dictionary
    self.cur_rm_pose[self.cur_rm_name] = []
    self.remove_cnt = -self.remove_cnt
    """
    Grip object
    x1,y1,x2,y2: Object outer border coordinates
    """
    #Gripping object program
    cmd1 = "ros2 run largemodel_arm KCF_Grap_Move"
    #KCF tracking positioning program
    cmd2 = "ros2 run largemodel_arm ALM_KCF_Tracker"
    subprocess.Popen(
        [
            "gnome-terminal",
            "--title=ALM_KCF_Tracker",
            "--",
            "bash",
            "-c",
            f"{cmd2}; exec bash",
        ]
    )
    time.sleep(5.0) #Wait for ALM_KCF_Tracker to start up
    subprocess.Popen(
        [
            "gnome-terminal",
            "--title=grasp_desktop",
            "--",
            "bash",
            "-c",
            f"{cmd1}; exec bash",
        ]
    )
    time.sleep(2.0)
    if self.stack_flag == True:
        self.get_logger().info('Publish the stack_step topic...')
        step_ = Int16()
        step_.data = self.step
        self.step_pub.publish(step_)
        self.step = self.step + 1
    #Publish outer border coordinate information topic data, ALM_KCF_Tracker node
will subscribe to this topic
    x1 = int(x1)
    y1 = int(y1)
    x2 = int(x2)
    y2 = int(y2)
    self.object_position_pub.publish(Int16MultiArray(data=[x1, y1, x2, y2]))

    while not self.grasp_obj_future.done():
        if self.interrupt_flag:
            self.check_close_grasp_obj()
            #self.pubSix_Arm(self.init_joints)  # Robotic arm retract
            #Arm.Arm_serial_servo_write6(90,150,12,20,90,30,1000)
            self.stop()
            return
        time.sleep(0.1)
    self.check_close_remove_obj()


    tmp_joint1 = 90 + self.remove_cnt*60 #30 150
    #Add the calculated joint values for placing removed color blocks to the
dictionary as the key value of self.cur_rm_pose[self.cur_rm_name]
    self.cur_rm_pose[self.cur_rm_name] = [tmp_joint1,50,40,20,90,30]
    self.get_logger().info(f"self.cur_rm_pose:{self.cur_rm_pose}")
    #Control robotic arm to move to specified posture to place removed color
blocks
    Arm.Arm_serial_servo_write6(tmp_joint1,50,40,20,90,140,2000)
    time.sleep(2.0)
    Arm.Arm_serial_servo_write6(tmp_joint1,50,40,20,90,30,2000)
    time.sleep(2.0)
    Arm.Arm_serial_servo_write6(90,120,10,10,90,30,2000)
    time.sleep(2.0)
    #Feedback to large model that color block removal is complete
    self.action_status_pub("remove_obj_done", x1=x1, y1=y1, x2=x2, y2=y2,
color=color)
```

KCF_Grap_Move and ALM_KCF_Tracker nodes have been analyzed in previous content, you can refer to section 2.4.3 in [17.AI Model-Text Version]-[Multimodal Large Model+Robotic Arm Gripping].

4.5. grasp_obj(x1, y1, x2, y2) Function

Source code path: `LargeModel_ws/src/largemodel/largemodel/action_service.py`

grasp_obj(x1, y1, x2, y2) function has been analyzed in previous content, you can refer to section 2.4.3 in [17.AI Model-Text Version]-[Multimodal Large Model+Robotic Arm Gripping].