

Multimodal Large Model + Color Blocks Back In Place (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, first input the command to remember the current positions of color blocks in the terminal. After the program finishes recording, scramble the positions of the color blocks, then input the command to return the color blocks on the desktop to their original positions. After thinking, the large model will plan an action list, and the program will control the robotic arm to complete the color block return task according to the action functions in the action list.

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 largemode1 largemode1_control.launch.py text_chat_mode:=True
```

Then open a second terminal and input the following command:

```
ros2 run text_chat text_chat
```

Now input the command to remember the current positions of color blocks in the terminal. You can refer to the following example:

Remember the current positions of the color blocks

```
user input: 记住色块现在的位置
okay让我想一想... -[INFO] [1764400308, 613808328] [text_chat_node]: 决策层AI规划:1. 调用`seewhat()`观察环境;
2. 重新调用`seewhat()`观察环境, 然后判断哪个色块是顶部色块;
3. 如果环境中只有两堆及两堆以上色块, 根据是否为单独色块, 选择执行以下内容, 对每堆色块进行计算, 这里每次只能输出一个动作函数;
   - 如果是单独色块, 那么把色块的外边框坐标传入函数`compute_pose(x1, y1, x2, y2, 'name')`, 其中`name`表示色块的颜色, 取值为`red`、`blue`、`green`或者`yellow`;
   - 如果不是单独色块, 那么把顶部色块的外边框坐标传入函数`compute_pose_order(x1, y1, x2, y2, 'name', 'order')`, 其中`name`表示色块的颜色, 取值为`red`、`blue`、`green`或者`yellow`; `order`表示色块排列由上到下的排列顺序, 取值为从上到下每一层色块颜色的首字母组合起来的字符串, 比如从上到下的色块的排列顺序是: red, blue, green, yellow, 那么`order`的取值就为`rbgy`。以此类推;
4. 如果只有一堆色块, 那么把顶部色块的外边框坐标传入函数`compute_pose_order(x1, y1, x2, y2, 'name', 'order')`, 其中`name`表示色块的颜色, 取值为`red`、`blue`、`green`或者`yellow`; `order`表示色块排列由上到下的排列顺序, 取值为从上到下每一层色块颜色的首字母组合起来的字符串, 比如从上到下的色块的排列顺序是: red, blue, green, yellow, 那么`order`的取值就为`abgr`。以此类推;
-[INFO] [1764400311, 3888902541] [text_chat_node]: "action": ["seewhat()"], "response": "好呀, 我现在就去看看桌面上的色块们都在哪里呢~";
-[INFO] [1764400317, 628307383] [text_chat_node]: "action": ["seewhat()"], "response": "我看到桌面上有三个色块堆了, 红色在黄色上面, 蓝色和绿色是单独的, 让我仔细记一下它们的位置";
-[INFO] [1764400325, 495346264] [text_chat_node]: "action": ["compute_pose_order(90, 305, 190, 405, 'red', 'ry')"], "response": "我先记住红色色块堆的位置, 它在黄色上面, 顺序是红黄";
-[INFO] [1764400330, 703854914] [text_chat_node]: "action": ["compute_pose(270, 315, 340, 385, 'blue')"], "response": "蓝色色块是单独的, 我也把它位置记下来啦~";
-[INFO] [1764400343, 591951080] [text_chat_node]: "action": ["compute_pose(400, 325, 470, 395, 'green')"], "response": "绿色色块也单独放着, 我已经把它位置记好啦~";
-[INFO] [1764400351, 079633306] [text_chat_node]: "action": ["finishtask()"], "response": "所有色块的位置都记好了, 红色在黄色上面, 蓝色和绿色是单独的, 我已经把它们的位置都存起来了~"
```

After recording the color block positions, input the end task command in the terminal:

```
End the position recording task
```

Then you can scramble the positions of the color blocks. After scrambling, input the color block return command in the terminal. You can refer to:

```
Return the blocks on the desktop to their original positions
```

```

user input: 把桌面上方块放回到原来的位置
okay let me think for a moment.
[INFO] [1764400625.599131872] [text_chat_node]: 决策层AI规划x1，首先，调用`color_back_to_origin()`函数，控制色块归位。
2. 调用 `check_remove(color)` 函数，检查移除列表中是否存在目标色块（其中，`color` 为为目标色块的颜色，取值范围为 red、blue、green 或 yellow）。
3. 若移除列表中存在目标色块，执行以下操作：
    (1) 调用 `grasp_from_rm_list(color)` 函数，在移除列表中取该目标色块（`color` 参数含义同上）；
    (2) 将目标色块的颜色作为参数传入 `return_to_origin(color)` 函数，例如目标色块为红色时，调用 `return_to_origin(red)`（`color` 参数取值范围同上）。
4. 若移除列表中不存在目标色块，需执行以下步骤：
    (1) 调用 `seewhat()` 函数，找到当前需要归位的目标色块的上面有没有其他物体；
    (2) 调用 `seewhat()` 函数，判断当前需要归位的目标色块的上面有没有别的方块呢~；
    (3) 若当前需要归位的目标色块顶部没有其他色块时，执行以下步骤：
        ①调用 `grasp_obj(x1, y1, x2, y2)` 函数取当前需要归位的目标色块（其中，` $(x_1, y_1, x_2, y_2)$ ` 为当前需要归位的目标色块的顶部的边框坐标）；
        ②将目标色块的颜色作为参数传入 `return_to_origin(color)` 函数（`color` 参数取值范围同上）。
    (4) 若当前需要归位的目标色块顶部有其他色块时，执行以下步骤：
        ①调用 `remove_obj(x1, y1, x2, y2, color)` 函数移除该色块（其中，` $(x_1, y_1, x_2, y_2)$ ` 为该色块顶部的边框坐标，`color` 为该色块的颜色，取值范围为 red、blue、green 或 yellow）；
        ②调用 `seewhat()` 函数，观察环境；
        ③若当前需要归位的目标色块的上方没有其他色块，调用 `grasp_obj(x1, y1, x2, y2)` 函数取目标色块（其中，` $(x_1, y_1, x_2, y_2)$ ` 为目标色块顶部的边框坐标）；
        ④将目标色块的颜色作为参数传入 `return_to_origin(color)` 函数（`color` 参数取值范围同上）。
[INFO] [1764400627.509251997] [text_chat_node]: action: ['color_back_to_origin'], response: "好呀，我这就开始把桌面上的方块放回原来的位置啦~"
[INFO] [1764400629.666590011] [text_chat_node]: action: ['check_remove("green")'], response: "我正在检查移除列表里有没有绿色方块呢，稍等一下哦~"
[INFO] [1764400631.564086420] [text_chat_node]: action: ['seewhat()'], response: "我看到绿色方块的位置，看看它上面有没有别的方块呢~"
[INFO] [1764400637.079474364] [text_chat_node]: action: ['seewhat()'], response: "我看到绿色方块上面没有其他物体，可以安全地夹取它啦~"
[INFO] [1764400643.259706518] [text_chat_node]: action: ['grasp_obj(405, 253, 485, 331)'], response: "我找到绿色方块啦，现在准备把它夹起来放回原位~"
[INFO] [1764400660.983629961] [text_chat_node]: action: ['return_to_origin("green")'], response: "绿色方块已经成功放回原来的位置啦，任务完成得漂漂亮亮的~"
[INFO] [1764400673.349340900] [text_chat_node]: action: ['check_remove("blue")'], response: "我正在检查移除列表里有没有蓝色方块呢，稍等一下哦~"
[INFO] [1764400678.349706065] [text_chat_node]: action: ['seewhat()'], response: "我在观察蓝色方块的位置，看看它上面有没有别的方块呢~"
[INFO] [1764400685.783486011] [text_chat_node]: action: ['seewhat()'], response: "我看到蓝色方块上面没有其他物体，可以安全地夹取它啦~"
[INFO] [1764400695.170324356] [text_chat_node]: action: ['grasp_obj(90, 278, 180, 365)'], response: "我找到蓝色方块啦，现在准备把它夹起来放回原位~"
[INFO] [1764400713.170749470] [text_chat_node]: action: ['return_to_origin("blue")'], response: "蓝色方块已经成功放回原来的位置啦，任务完成得漂漂亮亮的~"
[INFO] [1764400723.811916742] [text_chat_node]: action: ['seewhat()'], response: "我正在检查移除列表里有没有黄色方块呢，稍等一下哦~"
[INFO] [1764400726.248457991] [text_chat_node]: action: ['seewhat()'], response: "我正在观察黄色方块的位置，看看它上面有没有别的方块呢~"
[INFO] [1764400732.608013012] [text_chat_node]: action: ['seewhat()'], response: "我在观察黄色方块的位置，看看它上面有没有黄色方块，需要先移除它才能夹取黄色方块啦~"
[INFO] [1764400739.489793839] [text_chat_node]: action: ['remove_obj(240, 305, 330, 390, "red")'], response: "我准备先把红色方块移除，这样就顺利夹取黄色方块啦~"
[INFO] [1764400772.056767437] [text_chat_node]: action: ['seewhat()'], response: "红色方块已经移除了，我现在确认一下黄色方块上方是否还有其他物体~"
[INFO] [1764400779.190021136] [text_chat_node]: action: ['grasp_obj(235, 325, 460)'], response: "黄色方块现在可以安全夹取啦，我这就把它拿起来放回原位~"
[INFO] [1764400893.298027925] [text_chat_node]: action: ['return_to_origin("yellow")'], response: "黄色方块已经成功放回原来的位置啦，所有方块都归位完成咯~"
[INFO] [1764400903.991302022] [text_chat_node]: action: ['check_remove("red")'], response: "我正在检查移除列表里有没有红色方块呢，稍等一下哦~"
[INFO] [1764400907.422547341] [text_chat_node]: action: ['grasp_from_rm_list("red")'], response: "我从移除列表中找到了红色方块，现在准备把它夹起来放回原位啦~"
[INFO] [1764400921.510202192] [text_chat_node]: action: ['return_to_origin("red")'], response: "红色方块已经成功放回原来的位置啦，所有任务都完成得漂漂亮亮的~"
[INFO] [1764400937.492499155] [text_chat_node]: action: ['finishtask()'], response: "所有方块都已成功放回原位啦，任务圆满完成！我今天可是超级贴心的小帮手呢~"

```

After the buzzer beeps once, the program starts gripping color blocks and returning them to their original positions according to the order recorded in the program. If there are other color blocks above the current color block that needs to be returned, the upper color blocks will be removed first, then the target block will be gripped and returned to its original position.

3. Task Planning

3.1. Record Current Positions of Color Blocks

1. Call `seewhat()` to observe the environment;
2. Call `seewhat()` again to observe the environment, then determine which color block is the top block;
3. If there are two or more piles of color blocks in the environment, based on whether it's a single block, choose to execute the following content, calculate for each pile of blocks, here only one action function can be output at a time:
 - o If it's a single block, pass the outer border coordinates of the block to the function `compute_pose(x1, y1, x2, y2, 'name')`, where `name` represents the color of the block, values are `red`, `blue`, `green` or `yellow`;
 - o If it's not a single block, pass the outer border coordinates of the top block to the function `compute_pose_order(x1, y1, x2, y2, 'name', 'order')`, where `name` represents the color of the block, values are `red`, `blue`, `green` or `yellow`, `order` represents the stacking order from top to bottom, values are strings composed of the first letters of each layer's color from top to bottom, for example if the stacking order from top to bottom is: red, blue, green, yellow, then the value of `order` is `rbgy`, and so on.
4. If there's only one pile of blocks, pass the outer border coordinates of the top block to the function `compute_pose_order(x1, y1, x2, y2, 'name', 'order')`, where `name` represents the color of the block, values are `red`, `blue`, `green` or `yellow`, `order` represents the stacking order from top to bottom, values are strings composed of the first letters of each layer's color from top to bottom, for example if the stacking order from top to bottom is: red, blue, green, yellow, then the value of `order` is `rbgy`, and so on.

3.2. Color Blocks Return

1. First, call the `color_back_to_orin()` function to control color blocks to return to original positions.
2. Call the `check_remove(color)` function to check if the target color block exists in the removal list (where `color` is the color of the target block, value range is `red`, `blue`, `green` or `yellow`).
3. If the target color block exists in the removal list, execute the following operations:
 - (1) Call the `grasp_from_rm_list(color)` function to grip the target color block from the removal list (meaning of `color` parameter is the same as above);
 - (2) Pass the color of the target block as parameter to the `return_to_orin(color)` function, for example when the target block is red, call `return_to_orin(red)` (value range of `color` parameter is the same as above).
4. If the target color block does not exist in the removal list, follow these steps to return the block to its original position (only output one action function at a time):
 - (1) Call the `seewhat()` function to find the current target block that needs to be returned;
 - (2) Call the `seewhat()` function to determine if there are other objects above the current target block that needs to be returned;
 - (3) If there are no other color blocks on top of the current target block that needs to be returned, execute the following steps:
 - ① Call the `grasp_obj(x1, y1, x2, y2)` function to grip the current target block that needs to be returned (where `(x1, y1, x2, y2)` are the top surface border coordinates of the current target block that needs to be returned);
 - ② Pass the color of the target block as parameter to the `return_to_orin(color)` function (value range of `color` parameter is the same as above).
 - (4) If there are other color blocks on top of the current target block that needs to be returned, execute the following steps:
 - ① Call the `remove_obj(x1, y1, x2, y2, color)` function to remove that block (where `(x1, y1, x2, y2)` are the top surface border coordinates of that block, `color` is the color of that block, value range is `red`, `blue`, `green` or `yellow`);
 - ② Call the `seewhat()` function to observe the environment;
 - ③ If there are no other color blocks above the current target block that needs to be returned, call the `grasp_obj(x1, y1, x2, y2)` function to grip the target block (where `(x1, y1, x2, y2)` are the top surface border coordinates of the target block);
 - ④ Pass the color of the target block as parameter to the `return_to_orin(color)` function (value range of `color` parameter is the same as above)

4. Core Code Analysis

4.1. compute_pose_order Record Stacked Color Block Position

Source code path: `LargeModel_ws/src/largemode1/largemode1/action_service.py`

```
#Parameter description
#x1,y1,x2,y2: Top outer border coordinates of the top color block, x1,y1
represent
#name: Color of the top color block
#order: Represents the stacking order list from top to bottom, for example if the
stacked colors from top to bottom are red and yellow, then the passed order is
ry
def compute_pose_order(self,x1,y1,x2,y2,name,order):
    #Get current color block parameters
```

```

cur_name = name.strip("'\\"")
color_order = order.strip("'\\"")
#Create a list to store the stacking order of color blocks from top to
bottom
order_list = list(color_order[1:])
self.get_logger().info(f"order_list:{order_list}")
self.cur_name = cur_name
self.cur_pose[self.cur_name] = []
#Start position calculation program
cmd1 = "ros2 run largemodel_arm Record_pose"
proc = subprocess.Popen(
    [
        "gnome-terminal",
        "--title=compute_pose",
        "--",
        "bash",
        "-c",
        f"{cmd1}; exec bash",
    ]
)
self.pid_num = proc.pid
self.get_logger().info(f"self.pid_num:{self.pid_num}")
time.sleep(5.0)
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.compute_pose_future.done():
    if self.interrupt_flag:
        self.check_close_compute_pose()
        Arm.Arm_serial_servo_write6(90,150,12,20,90,30,1000)
        self.stop()
        return
    time.sleep(0.1)

result = self.compute_pose_future.result()
self.check_close_compute_pose()
if not self.interrupt_flag:
    if result.data == "compute_pose_done":
        for i in range(0,len(order_list)):
            #Here we consider that xy directions are consistent with the top
            #block, only height changes, so just need to change z value, decreasing by 2.8cm
            #each time, approximately the height of one color block
            x = self.cur_pose[cur_name][0]
            y = self.cur_pose[cur_name][1]
            z = self.cur_pose[cur_name][2] - (i+1)*0.028
            if order_list[i] == 'r' or order_list[i] == 'R':
                self.cur_pose['red'] = []
                self.cur_pose['red'].append(x)
                self.cur_pose['red'].append(y)
                self.cur_pose['red'].append(z)
            elif order_list[i] == 'b' or order_list[i] == 'B':
                self.cur_pose['blue'] = []
                self.cur_pose['blue'].append(x)
                self.cur_pose['blue'].append(y)
                self.cur_pose['blue'].append(z)

```

```

        elif order_list[i] == 'y' or order_list[i] == 'Y':
            self.cur_pose['yellow'] = []
            self.cur_pose['yellow'].append(x)
            self.cur_pose['yellow'].append(y)
            self.cur_pose['yellow'].append(z)
        elif order_list[i] == 'g' or order_list[i] == 'G':
            self.cur_pose['green'] = []
            self.cur_pose['green'].append(x)
            self.cur_pose['green'].append(y)
            self.cur_pose['green'].append(z)

        self.get_logger().info(f"self.cur_pose:{self.cur_pose}")
        #Feedback to large model that recording stacked color block
positions is complete
        self.action_status_pub("compute_pose_order_done", x1=x1, y1=y1,
x2=x2, y2=y2, name=name, order=order)
    else:
        self.action_status_pub("compute_pose_failed", x1=x1, y1=y1, x2=x2,
y2=y2)

```

4.2. color_back_to_orin Publish Color Block Return Order

Source code path: LargeModel_ws/src/largemode1/largemode1/action_service.py

```

def color_back_to_orin(self):
    self.back_list = True
    #Buzzer beeps once
    Arm.Arm_Buzzer_On()
    time.sleep(0.5)
    Arm.Arm_Buzzer_Off()
    #Traverse the list storing color block positions in reverse order, feedback
to large model the current color block return instruction
    for key in reversed(self.cur_pose.keys()):
        if key == 'red':
            self.action_status_pub("send_red_back_to_orin")
        elif key == 'green':
            self.action_status_pub("send_green_back_to_orin")
        elif key == 'blue':
            self.action_status_pub("send_blue_back_to_orin")
        elif key == 'yellow':
            self.action_status_pub("send_yellow_back_to_orin")
    while not self.return_done:
        time.sleep(0.1)
    self.return_done = False
    self.action_status_pub("return_to_orin_done")
    self.back_list = False
"send_red_back_to_orin": "Robot feedback: Return the red block to its original
position",
"send_green_back_to_orin": "Robot feedback: Return the green block to its
original position",
"send_blue_back_to_orin": "Robot feedback: Return the blue block to its original
position",
"send_yellow_back_to_orin": "Robot feedback: Return the yellow block to its
original position"

```

4.3. check_remove(color) Check if Current Target Color Block Exists in Removal List

Source code path: LargeModel_ws/src/largemode1/largemode1/action_service.py

```
def check_remove(self,color):
    #Get current color block color
    check_color = color.strip("\\")

    #If current color block color can be found in removal list, then feedback to large model that current color block can be found in 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")

"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",
```

4.4. grasp_from_rm_list Grip Color Block from Removal List

Source code path: LargeModel_ws/src/largemode1/largemode1/action_service.py

```
def grasp_from_rm_list(self,color):
    #Get current color block color
    tar_color = color.strip("\\")

    #Find corresponding robotic arm posture value in removal dictionary based on current color
    tar_joints = self.cur_rm_pose.get(tar_color)
    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_write6(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 from removal list is complete
    self.action_status_pub("grasp_from_rm_list_done")
```