Color Recognition and Sorting with Voice Broadcast

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].

Orin board users can directly open the terminal and enter the tutorial commands to run. Jetson-Nano board users need to enter the docker container first, then enter the tutorial commands in the docker to start the program.

1. Function Description

The voice module broadcasts the current color block color and the robotic arm grabs and places it in the set position.

2. Startup and Operation

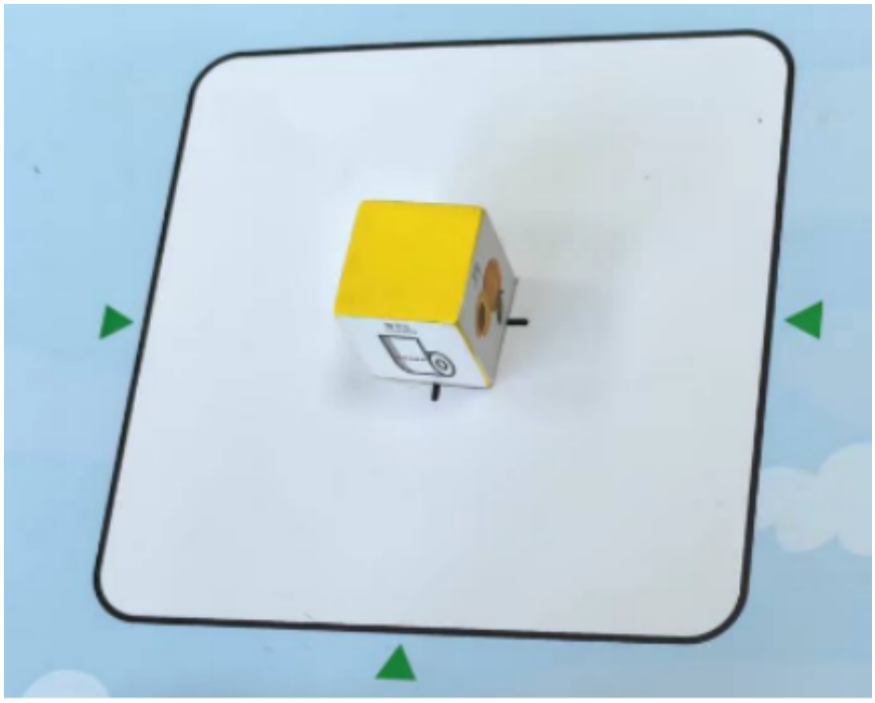
2.1. Startup

Open the terminal on the main board and enter the following command:

```
python3 ~/dofbot_voice/scripts/put_grab_broadcast.py
```

2.2. Operation Steps

After the program starts, the robotic arm automatically reaches the recognition position. Place the block on the cross stand. After the robotic arm correctly recognizes it 10 times continuously, the voice module will broadcast the recognized color, then automatically grab and place it in the corresponding color grid. Finally, it will broadcast "Placement complete". Taking the figure below as an example, the voice module will broadcast "I see Yellow", then grab and place it in the corresponding color grid. Note: If colors cannot be recognized or the target color cannot be tracked correctly, you may need to calibrate the HSV values in the program. Please refer to [28.Visual Basic Course] -> [2.Color Calibration].



3. Core Code Analysis

Jetson-Nano users need to enter the docker container to view

`Source code path: ~/dofbot_voice/scripts/put_grab_broadcast.py`

```
#Import voice broadcast color block sorting library, this library is located at
dofbot_pro/dofbot_color_grab/scripts/speech_color_sorting.py
from speech_color_sorting import speech_color_sorting

# Create instance
sorting = speech_color_sorting()
#Call function, pass image and color HSV threshold for color block color
recognition
img = sorting.Sorting_grap(img, color_hsv)

#Sorting_grap function is located in speech_color_sorting library
def Sorting_grap(self, img, color_hsv):
    # Standardize input image size
    self.image = cv.resize(img, (640, 480))
    # Get recognition result
    msg = {}
    if self.status == 'waiting':
        # Traverse color channels, get recognizable results
        for key, value in color_hsv.items():
            point = self.get_Sqaure(key, value)
            if point != None: msg["name"] = key
        if len(msg) == 1:
            self.num += 1
            # Every time continuous recognition reaches 10 times and motion
status is waiting, execute grabbing task
            if self.num % 10 == 0 and self.status == 'waiting':

                self.status = "Runing"
                self.arm.Arm_Buzzer_On(1)
                sleep(0.5)
                # Start grabbing thread
                threading.Thread(target=self.sorting_run, args=
(msg['name'],)).start()
                self.num = 0
    return self.image
#sorting_run function is located in speech_color_sorting library
def sorting_run(self, name):
    #According to the value of name, broadcast "what color is recognized", then
execute self.sorting_move and pass the placement position
    if name == "red" :
        mySpeech.void_write(88)
        time.sleep(1.0)
        mySpeech.void_write(46)
        # print("red")
        self.sorting_move(self.robot.P_RED)
        mySpeech.void_write(81)
        # Grabbing complete
        self.status = 'waiting'
    if name == "blue":
        mySpeech.void_write(88)
        time.sleep(1.0)
```

```
            mySpeech.void_write(48)
            # print("blue")
            self.sorting_move(self.robot.P_BLUE)
            mySpeech.void_write(81)
            # Grabbing complete
            self.status = 'waiting'
        if name == "green" :
            mySpeech.void_write(88)
            time.sleep(1.0)
            mySpeech.void_write(49)
            # print("green")
            self.sorting_move(self.robot.P_GREEN)
            mySpeech.void_write(81)
            # Grabbing complete
            self.status = 'waiting'
        if name == "yellow" :
            mySpeech.void_write(88)
            time.sleep(1.0)
            mySpeech.void_write(47)
            # print("yellow")
            self.sorting_move(self.robot.P_YELLOW)
            mySpeech.void_write(81)
            # Grabbing complete
            self.status = 'waiting'
```