Apriltag recognition grabbing blocks

1. apriltag identification instructions

The default format used for apriltag tag code recognition is TAG36H11 image, which comes with relevant tag codes and is affixed to the blocks.

2. Map placement

Place the block labeled with the code on the cross in the recognition area.

3. About code

Code path： ~/jetcobot_ws/src/jetcobot_grasp/scripts/3_apriltag_recognition_grasp.py

~/jetcobot_ws/src/jetcobot_utils/src/jetcobot_utils/grasp_controller.py

The function of controlling the movement and grasping of the robotic arm.

```
def grasp_run(self,tagId):
        self.graspController.goBoxCenterlayer1Pose()
        self.graspController.close_gripper(1.5)
        self.graspController.goColorOverPose()
        if tagId == '1':
            self.graspController.goApriltag1fixedPose()
        elif tagId == '2':
            self.graspController.goApriltag2fixedPose()
        elif tagId == '3':
            self.graspController.goApriltag3fixedPose()
        elif tagId == '4':
            self.graspController.goApriltag4fixedPose()
        else:
            self.graspController.init_watch_pose()
            self.status = 'waiting'
            return
        time.sleep(1)
        self.graspController.open_gripper(1)
        self.graspController.rise_gripper(1)
        self.graspController.init_watch_pose()
        self.status = 'waiting'
```

Identify the position coordinates of the cross in the middle of the area.

```
def goBoxCenterlayer1Pose(self):
        coords = [220, 0, 120, -175, 0, -45]
        self.go_coords(coords, 3)
```

The coordinate value of the location where the tag code is placed.

If the placement position coordinates are inaccurate, you can modify this coordinate value appropriately.

```
# Apriltag label position
    def goApriltag1fixedPose(self, layer=1):
        coords = [-60, 170, 105+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)

    def goApriltag2fixedPose(self, layer=1):
        coords = [10, 170, 105+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)

    def goApriltag3fixedPose(self, layer=1):
        coords = [75, 170, 105+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)

    def goApriltag4fixedPose(self, layer=1):
        coords = [140, 170, 110+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)
```

4. Start program

Start roscore

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.
- Then, run roscore

```
sh ~/start_docker.sh
roscore
```

- If you are using Jetson Nano board. You need to enter the following command directly.

```
roscore
```

Start the program

Open a new terminal.

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.

```
sh ~/start_docker.sh
rosrun jetcobot_grasp 3_apriltag_recognition_grasp.py
```

- If you are using Jetson Nano board. You need to enter the following command directly.

```
rosrun jetcobot_grasp 3_apriltag_recognition_grasp.py
```

5. Experimental results

After the program runs, the camera will detect whether there is a tag code in the recognition area.

If there is a tag code, grab the accumulated wooden block from the cross in the middle of the recognition area, and place the corresponding ID of the accumulated wooden block in the corresponding area. After the placement is completed, resume the recognition state and place the building block again.

Note: If there are two building blocks with the same ID number, it is necessary to clean up the blocks in the numerical area first to avoid placement conflicts.