

Gesture Grasp and Release Objects

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

1. Introduction

The gesture grasp and release objects function is based on gesture recognition, adding the capability of specific gestures to control the robotic arm. When there is gesture 5 in the camera image, the robotic arm will move to a specific location to grasp objects. When there is gesture 1 in the camera image, the robotic arm will place objects at a specific location.

Recognizable gestures include: [One, Five], a total of 2 categories.

2. Launch

- Open the desktop terminal and enter the following command to start the program

```
ros2 run dofbot_pro_mediapipe 17_GestureGrasp
```

At this time, place your hand in the camera frame and make gesture 5, then the robotic arm will move forward to grasp the object.



After grasping the object, when gesture 1 is recognized, the object will be placed at the upper left position.



Press the q key in the image or press Ctrl+c in the terminal to exit the program.

3. Source Code

Code path:

```
# Jetson-Nano users need to enter the docker container to view
~/dofbot_pro_ws/src/dofbot_pro_mediapipe/dofbot_pro_mediapipe/17_GestureGrasp.py
```

```
#!/usr/bin/env python3
# encoding: utf-8
import cv2 as cv
import time
import sys
import signal
import rclpy
from rclpy.node import Node
from dofbot_utils.fps import FPS
from dofbot_utils.robot_controller import Robot_Controller
from dofbot_utils.GestureRecognition import handDetector
from Arm_Lib import Arm_Device
import threading

class GestureActionNode(Node):
    def __init__(self):
        super().__init__('gesture_action')
        self.hand_detector = handDetector(detectorCon=0.75)
        self.pTime = 0

        # Define the state of grabbing blocks
        self.one_grabbed = 0
        self.two_grabbed = 0
        self.three_grabbed = 0
        self.four_grabbed = 0

        self.block_num = 0

        # Define the number of gesture recognitions
        self.Count_One = 0
        self.Count_Two = 0
        self.Count_Three = 0
        self.Count_Four = 0
        self.Count_Five = 0

        self.arm = Arm_Device()
        self.move_state = False
        self.fps = FPS()
        self.robot = Robot_Controller()
        self.grap_joint = self.robot.get_gripper_value(1)
        self._joint_5 = self.robot.joint5
        self.arm.Arm_serial_servo_write6_array(self.robot.P_LOOK_AT, 1000)

        # Initialize video capture device
        self.cap = cv.VideoCapture(0, cv.CAP_V4L2)
        self.cap.set(cv.CAP_PROP_FRAME_WIDTH, 640)
        self.cap.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
        if not self.cap.isOpened():
```

```

self.get_logger().error("Error: Could not open video device.")
rcipy.shutdown()

def process(self, frame):
    frame, lmList = self.hand_detector.findHands(frame, draw=False)
    if len(lmList) != 0:
        gesture = self.hand_detector.get_gesture()
        # print("gesture = {}".format(gesture))

        if gesture == 'One':
            cv.putText(frame, gesture, (250, 30), cv.FONT_HERSHEY_SIMPLEX,
0.9, (0, 255, 0), 1)
            self.Count_One += 1
            self.Count_Two = 0
            self.Count_Three = 0
            self.Count_Four = 0
            self.Count_Five = 0
            if self.Count_One >= 5 and not self.move_state:
                self.move_state = True
                self.Count_One = 0
                print("start arm_ctrl_threading = {}".format(gesture))
                task = threading.Thread(target=self.arm_ctrl_threading,
name="arm_ctrl_threading", args=(gesture,))
                task.setDaemon(True)
                task.start()

            elif gesture == 'Five':
                cv.putText(frame, gesture, (250, 30), cv.FONT_HERSHEY_SIMPLEX,
0.9, (0, 255, 0), 1)
                self.Count_Five += 1
                self.Count_One = 0
                self.Count_Two = 0
                self.Count_Three = 0
                self.Count_Four = 0
                if self.Count_Five >= 5 and not self.move_state:
                    self.move_state = True
                    self.Count_Five = 0
                    print("start arm_ctrl_threading = {}".format(gesture))
                    task = threading.Thread(target=self.arm_ctrl_threading,
name="arm_ctrl_threading", args=(gesture,))
                    task.setDaemon(True)
                    task.start()

        self.fps.update_fps()
        self.fps.show_fps(frame)
        return frame

def arm_ctrl_threading(self, gesture):
    print("arm_ctrl_threading gesture = {}".format(gesture))
    if gesture == 'One':
        self.arm.Arm_serial_servo_write6_array([163, 111, 0, 53, 90, 135],
1000)

        time.sleep(1.5)
        self.arm.Arm_serial_servo_write(6, 30, 500)
        time.sleep(0.8)
        self.arm.Arm_serial_servo_write6_array([90, 164, 18, 0, 90, 135],
1000)

```

```

        time.sleep(1.2)

    elif gesture == 'Five':
        self.joints = [90, 35, 65, 0, 90, 30]
        # Release clamping jaws 松开夹爪
        self.arm.Arm_serial_servo_write(6, 30, 500)
        time.sleep(0.5)
        # Move to object position 移动至物体位置
        self.arm.Arm_serial_servo_write6_array(self.joints, 1000)
        time.sleep(1.2)
        # Grasp and clamp the clamping claw进行抓取,夹紧夹爪
        self.arm.Arm_serial_servo_write(6, 135, 500)
        time.sleep(0.6)
        self.arm.Arm_serial_servo_write6_array([90, 164, 18, 0, 90, 135],
1000)

        time.sleep(1.2)
        self.move_state = False

    def run(self):
        while rclpy.ok():
            ret, frame = self.cap.read()
            if not ret:
                self.get_logger().error("Error: Could not read frame from video
device.")
                break

            action = cv.waitKey(1) & 0xFF
            frame = self.process(frame)
            if action == ord('q'):
                break
            cv.imshow('frame', frame)
        self.cap.release()
        cv.destroyAllWindows()

def main(args=None):
    rclpy.init(args=args)
    gesture_action_node = GestureActionNode()
    try:
        gesture_action_node.run()
    except KeyboardInterrupt:
        pass
    finally:
        gesture_action_node.cap.release()
        cv.destroyAllWindows()
        rclpy.shutdown()

if __name__ == '__main__':
    main()

```

