ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO

Môn học: Robotics

ĐỀ TÀI:

ĐIỀU KHIỂN ROBOT DI CHUYỂN THEO NGƯỜI

Giảng viên hướng dẫn: **TS. Lê Nguyên Khôi**

TS. Tạ Việt Cường

Nguyễn Đình Tuân

Sinh viên: **Trần Đăng Anh, 17020590**

Trần Duy Việt, 16021432

Hà Nội, 1 tháng 12 năm 2020

Mục lục

I. GIỚI THIỆU.

Ngày nay, IOT đã thúc đẩy quá trình phát triển của hệ thống tương tác người - máy rất nhanh. Để các hệ thống có thể theo kịp sư phát triển ngày, hệ thống ROS ra đời nhằm cung cấp khả năng xử lý dữ liệu cho robot nhanh chóng. Do đó, việc biết sử dụng ROS sẽ giúp ta rất nhiều trong quá trình tiếp cận và lập trình cho chúng.

II. BÀI TOÁN.

1. Đề bài

- Lập trình robot di chuyển theo người.

2. Yêu cầu bài toán.

- Robot phải hoạt động và bám theo hành động di chuyển của vật chủ.
- Robot phải di chuyển linh hoạt, không bị giật khấc.

III. XỬ LÝ VẤN ĐỀ.

1. Tiếp cận bài toán.

a. Mục tiêu chung.

- Robot cần phải di chuyển được.
- Robot có thể nhận diện được người.
- Robot di chuyển theo vật chủ.

b. Môi trường phát triển.

- ROS là điều kiện tiên quyết
- ROS cần chạy trên máy chủ Linux để tối đa tốc độ.
- Thư viện openCV để xử lý ảnh

- Yolo: Model detecting

2. Tiền xử lý.

a. Cài đặt môi trường.

- Đối tượng: ROBOT Turtlebot3.
- Trên robot đã được cài sẵn
- Hiện tại, ROS đang có 3 phiên bản là :
    + Kinetic - linux : 16.04.
    + Melodic - linux: 18.04.
    + Neotic - linux: 20.04.16

Link: http://wiki.ros.org/ROS/Installation

- Ta cài đặt ROS-Melodic vì:
    + Hiện tại Robot đã được cài sẵn ROS-Melodic

```
~/ sudo apt install ros-melodic-desktop-full
```

- Cài đặt tool catkin.
- Cài đặt python3.
- Cài đặt thư viện tensorflow.

b. Kết nối robot.

- Ssh server robot(chú ý: phải dùng chung qua một wifi)
- mở cmd và gõ:

```
~/>> ifconfig
```

-> Để xem ip của máy chủ cá nhân

- Tiếp tục gõ:

```
~/ nano .bashrc.
```

- Lúc này, cửa sổ mở lên. Ta sẽ chỉnh ROSMASTERURI về máy chủ cá nhân

- Sau đó, gõ:

```
~/ source .bashrc.
```

3. Giải quyết bài toán

a. Xử lý chung.

- Tạo ra file: follow.py

```
# Xử lý ảnh với imageai
from imageai.Detection import ObjectDetection
import os
import cv2
```

- Khi báo class Twist (dùng để vector hóa chuyển động Robot)

```
from geometry_msgs.msg import Twist
import roslib
```

- Khai báo rospy, numpy

```
import rospy
import numpy as np
```

- Khai báo class Follow()

```
class Follow():

   def __init__(self):
       os.system("clear")
```

```
        execution_path = os.getcwd()
        # Init node image_view_detect
        rospy.init_node("img_detect_following")
        detector = ObjectDetection()
        detector.setModelTypeAsYOLOv3()
        custom_objects = detector.CustomObjects(person=True)
        detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
        detector.loadModel(detection_speed="fastest")

        # publish into cmd vel
        self.pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
             self.sub  =  rospy.Subscriber('/raspicam_node/image/compressed',
    CompressedImage, self.callback)
        self.vel = Twist()
        self.rate = rospy.Rate(10)

        self.img_root_size = [640, 480]
        self.frame = 1

        self.input_image_path = "./webCamImage.jpg"
        self.output_image_path = "./outputImage.jpg"
        self.min_prob = 20

        # time sleep ps
        self.rate_sleep = rospy.Rate(10)
        self.cv_bridge = cv_bridge.CvBridge()

        while not rospy.is_shutdown():
            # check, frame = vs.read()
            # img = msg.data
            # np_arr = np.fromstring(img, np.uint8)
            # img_np = cv2.imdecode(np_arr, 1)

            # cv2.imwrite(input_image_path, frame)
            cv2.imwrite(self.input_image_path, self.frame)
                                                        detections      =
```

```
detector.detectCustomObjectsFromImage(custom_objects=custom_objects,

input_image=os.path.join(execution_path,

self.input_image_path),

output_image_path=os.path.join(execution_path,

self.output_image_path),

minimum_percentage_probability=self.min_prob)
            print("DA", detections)

            if len(detections) != 0:
                x1 = detections[0]["box_points"][0]
                y1 = detections[0]["box_points"][1]
                x2 = detections[0]["box_points"][2]
                y2 = detections[0]["box_points"][3]
                # print("(", x1, ", ", y1, ")")
                # print("(", x2, ", ", y2, ")")
                image_child = [x1, y1, x2, y2]
                self.listenAndMove(self.img_root_size, x1,y1,x2,y2)
                print("--------------------------------")
            else:
                print("Not found any person")
                self.stop()
                print("--------------------------------")

            img = cv2.imread(self.output_image_path)
            cv2.imshow('frame', img)
            # self.rate_sleep.sleep()
            key = cv2.waitKey(1)
            if key == ord("q"):
                self.stop()
                break
            self.rate.sleep()
```

```
        self.stop()

    def callback(self, data):
        # Handle Data
            #    self.frame  =  self.cv_bridge.compressed_imgmsg_to_cv2(data,
desired_encoding='bgr8')
        img = data.data
        np_arr = np.fromstring(img, np.uint8)
        self.frame = cv2.imdecode(np_arr, 1)
        # cv2.imshow('cv_img', img_np)

    def listenAndMove(self, img_root_size, x1,y1,x2,y2):

        person_center = (x1 + x2) / 2
        if person_center > (img_root_size[0] / 2 + 40):
            print("go Right")
            self.vel.angular.z = -0.5
        elif person_center < (img_root_size[0] / 2 - 40):
            print("go Left")
            self.vel.angular.z = 0.5
        else:
            self.vel.angular.z = 0.0

              if  y2  >  img_root_size[1]  or  (x2  -  x1)  *  (y2  -  y1)  >
img_root_size[0]*img_root_size[1]*0.5:
            print("Stop")
            self.vel.linear.x = 0.0
        else:
            self.vel.linear.x = 0.5

        self.pub.publish(self.vel)

    def stop(self):
        self.vel.linear.x = 0
        self.vel.angular.z = 0
```

```
            self.pub.publish(self.vel)
            rospy.sleep(1)

    if __name__ == '__main__':
        try:
            Follow()

        except rospy.ROSInterruptException:
            rospy.loginfo("Action terminated")

    # Calculator vector by img
```

Ta sẽ phân tích các thành phần của file như sau:

1. Hàm khai báo vào dọn dẹp biến môi trường cho hàm:

```
    os.system("clear")
    execution_path = os.getcwd()
```

2. Khởi tạo node trong rospy và đặt tên:

```
    rospy.init_node("img_detect_following")
```

3. Khai báo và cài đặt thuộc tính cho yolo để nhận diện persons:

```
    detector = ObjectDetection()
    detector.setModelTypeAsYOLOv3()
    custom_objects = detector.CustomObjects(person=True)
    detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
    detector.loadModel(detection_speed="fastest")
```

4. Khởi tạo Publisher để lắng nghe từ cmd_vel để điều khiển robot

```
    self.pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
```

5. Khởi tạo Subscriber để lấy data từ raspicam_node để lấy dữ liệu detect

```
    self.sub        =        rospy.Subscriber('/raspicam_node/image/compressed',
    CompressedImage, self.callback)
```

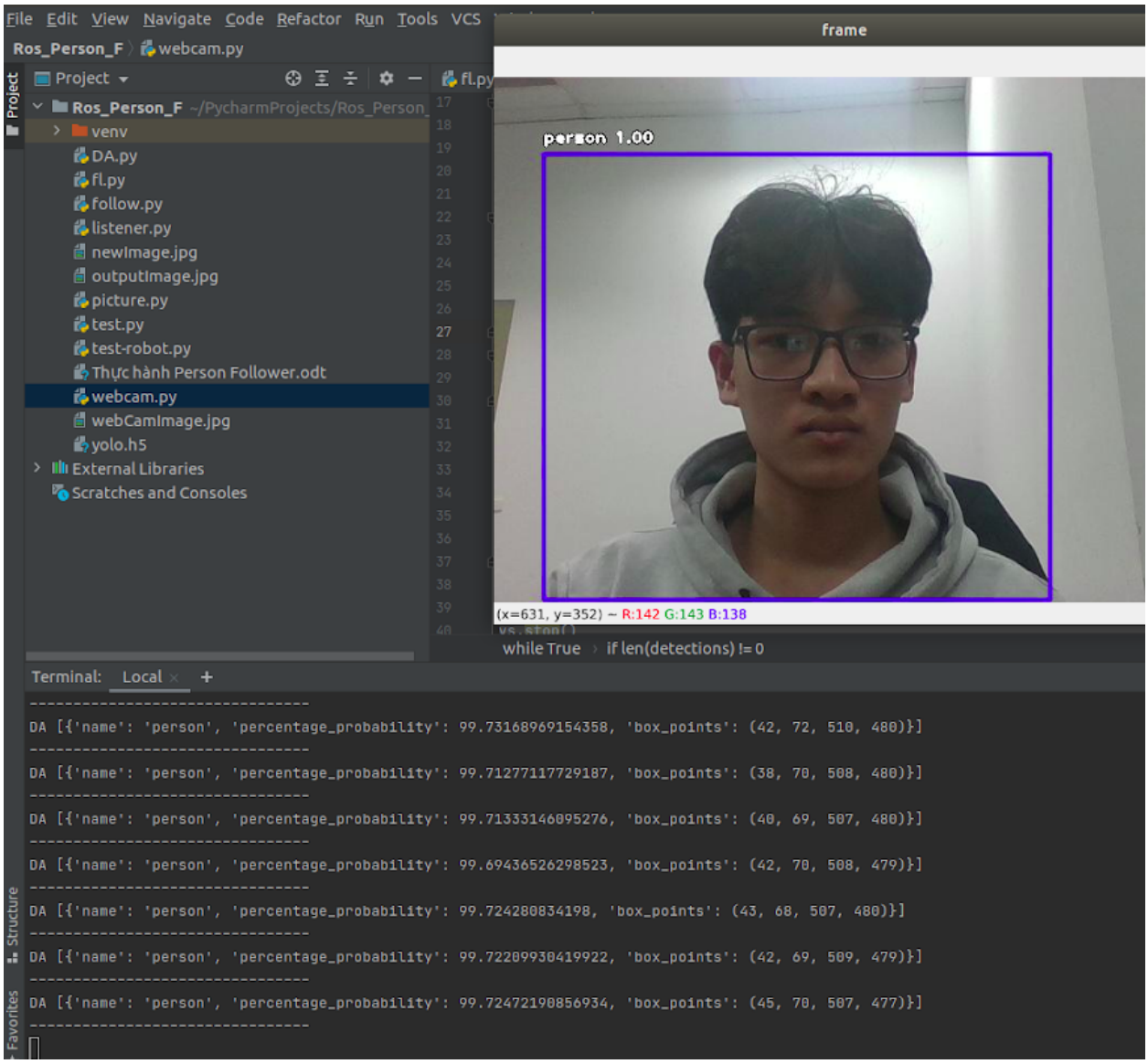
6. Khởi tạo tham số để bao gồm:

- Vel: vector để đưa robot di chuyển.
- Rate: Quay vòng thời gian listen từ node
- Frame: số khung hình.
- Input_image_path : Đầu vào hình ảnh.
- Output_image_path: Đầu ra hình ảnh
- Min_prob: Tham số detect.
- Rate_sleep: Tham số quay vòng dữ liệu của ros

```
    self.pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
    self.sub        =        rospy.Subscriber('/raspicam_node/image/compressed',
    CompressedImage, self.callback)
    self.vel = Twist()
    self.rate = rospy.Rate(10)

    self.img_root_size = [640, 480]
    self.frame = 1

    self.input_image_path = "./webCamImage.jpg"
    self.output_image_path = "./outputImage.jpg"
```

```
    self.min_prob = 20

    # time sleep ps
    self.rate_sleep = rospy.Rate(10)
```

7. Lắng nghe liên tục từ Ros , gửi dữ liệu qua detection để nhận diện persons nghe được qua rpicamera có phải là người không, nếu có sẽ trả về box detect với tham số dạng array Array(box_points [x1,y1,x2, y2]), Trong đó:
   - x1: hoành độ trên của box_points ,
   - y1: tung độ trên box_points ,
   - x2: hoành độ dưới box_points ,
   - y2: tung độ dưới box_points .



Ví dụ: (x1,y1,x2,y2) = (45, 70, 507, 477)

Tham số của box_point sẽ được gửi vào self.listenAndMove để xử lý di chuyển. Ở đây ta mặc định sẽ chỉ di chuyển theo đối tượng person đầu tiên (trái, trên cùng) - detections[0].

```
    cv2.imwrite(self.input_image_path, self.frame)
    detections =
    detector.detectCustomObjectsFromImage(custom_objects=custom_objects,

    input_image=os.path.join(execution_path,

    self.input_image_path),

    output_image_path=os.path.join(execution_path,

    self.output_image_path),

    minimum_percentage_probability=self.min_prob)
    print("DA", detections)

    if len(detections) != 0:
        x1 = detections[0]["box_points"][0]
        y1 = detections[0]["box_points"][1]
        x2 = detections[0]["box_points"][2]
        y2 = detections[0]["box_points"][3]
        # print("(", x1, ", ", y1, ")")
        # print("(", x2, ", ", y2, ")")
        image_child = [x1, y1, x2, y2]
        self.listenAndMove(self.img_root_size, x1,y1,x2,y2)
        print("--------------------------------")
    else:
        print("Not found any person")
        self.stop()
        print("--------------------------------")
```

8. Hàm listen listenAndMove sẽ định hướng vector để tác động di chuyển pi, trong đó:
   - Vel.angular: vector quay, ở đây ta chỉ quan tâm tham số Z.
   - Vel.linear: vector tiến, ở đây ta chỉ quan tâm tham số X.

- Sau khi xử lý được tham số trên, publisher sẽ binding data truyền vào cmd_vel để điều khiển robot

```
def listenAndMove(self, img_root_size, x1,y1,x2,y2):

    person_center = (x1 + x2) / 2
    if person_center > (img_root_size[0] / 2 + 40):
        print("go Right")
        self.vel.angular.z = -0.5
    elif person_center < (img_root_size[0] / 2 - 40):
        print("go Left")
        self.vel.angular.z = 0.5
    else:
        self.vel.angular.z = 0.0

        if y2 > img_root_size[1] or (x2 - x1) * (y2 - y1) >
img_root_size[0]*img_root_size[1]*0.5:
        print("Stop")
        self.vel.linear.x = 0.0
    else:
        self.vel.linear.x = 0.5

    self.pub.publish(self.vel)
```

9. Hàm self.stop() sẽ dừng di chuyển robot khi nó ở quá gần hoặc khi không lắng nghe từ publisher nữa.

```
def stop(self):
    self.vel.linear.x = 0
    self.vel.angular.z = 0
    self.pub.publish(self.vel)
    rospy.sleep(1)
```

10. Hàm callback() sẽ binding dữ liệu từ rpicamera và chuyển từng khung hình vào ảnh để xử lý ảnh trong thư viện cv2 được khai báo ở ban đầu.

```
def callback(self, data):
    # Handle Data
        #     self.frame   =   self.cv_bridge.compressed_imgmsg_to_cv2(data,
desired_encoding='bgr8')
    img = data.data
    np_arr = np.fromstring(img, np.uint8)
    self.frame = cv2.imdecode(np_arr, 1)
    # cv2.imshow('cv_img', img_np)
```

IV. KẾT QUẢ.

1. Xử lý di chuyển:

- Robot đã di chuyển theo chân người thành công.
- Robot di chuyển mượt và hiệu quả.

  https://photos.app.goo.gl/ZdUhDKaX896FkKXL7

  Link rút gọn: bom.to/Dok3OEsh

2. Phân công công việc.

- Trần Duy Việt : Nghiên cứu, debug, báo cáo. 50%
- Trần Đăng Anh : Nghiên cứu, code Node. 50%

V. THAM KHẢO.

- Sử dụng tài liệu tại : Roswiki

http://wiki.ros.org/

- Sử dụng tham khảo tài liệu điều khiển turtlebot3.

https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

Cám ơn các thầy :

TS. Lê Nguyên Khôi

TS. Tạ Việt Cường

Nguyễn Đình Tuân

Đã hướng dẫn chúng em trong việc làm bài tập này.