R550 control

轮趣科技Jetson nano ros小车控制；

prerequisite konwledge 先备知识

- 工控中上位机和下位机的基本概念
- 闭环控制的基本概念
- 汽车的阿克曼模型
- ros中publisher，topic，node，launch，message的基本知识
- STM32串口的概念
- 舵机驱动的基本方法

ros 2 θ and v 指令向底层的转变

ros的意义是让操作者直接对vehicle的位置环进行控制，而姿态环的控制交给底层的控制器（pixhawk4）或STM32，一般是一个反馈控制。

但是我们的算法设计中，基于车辆动力学模型（最简单的阿克曼转向模型）的积分是在训练中使用的。

但是对于该版本无人车，没有如pixhawk之类的控制器，且厂家已经在STM32写好了从速度+角速度转变为电机转速+前轮转向角的程序，因此我们的设计选择基于线速度和角速度。

如果你在学习ros的过程中记得经典的海龟节点，可能会记起来有一个Twist类型的话题，从ros api/Twist，我们可以查询到该话题控制的内容：

Raw Message Definition

```
# This expresses velocity in free space broken into its linear and angular parts.
Vector3  linear
Vector3  angular
```

Compact Message Definition

```
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular
```

可以看出，该话题中消息负责的是线速度和角速度，共6个量。

经过测试，我们能改变的是`twist.linear.x`和`twist.angular.z`两个量，分别对应速度和角速度。

注意，ros中控制速度的话题不止这一个，需要查阅加以区分。

Source Code 源代码

```
#!/usr/bin/env python
# coding=utf-8
```

```
import rospy
from geometry_msgs.msg import Twist
import sys, select, termios, tty


# define default speed
v_linear_x = 0
v_linear_y = 0
v_angular = 0


# get keyboard input
# from Willow Garage, Inc.
def getKey():
    tty.setraw(sys.stdin.fileno())
    r_list, _, _ = select.select([sys.stdin], [], [], 0.1)
    if r_list:
        key = sys.stdin.read(1)
    else:
        key = ''

    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
    return key


# show info of speed and angular
def vel_info(speed, turn, angular):
    return "currently:\tlinear x %s\tlinear y %s\t angular z %s" % (speed, turn,
angular)


# main
if __name__=="__main__":
    # get keyboard info
    settings = termios.tcgetattr(sys.stdin)

    # init ros node
    rospy.init_node('turtlebot_teleop')

    # topic name: node + cmd_vel
    publisher = rospy.Publisher('~cmd_vel', Twist, queue_size=10)

    try:
        print("keyboard control connected...")
        print(vel_info(v_linear_x, v_linear_y, v_angular))

        while(1):
            key = getKey()

            if key == 'w':
                v_linear_x += 0.1
            elif key == 's':
```

```
                v_linear_x -= 0.1
            elif key == 'a':
                v_linear_y += 0.1
            elif key == 'd':
                v_linear_y -= 0.1
            elif key == 'o':
                v_angular += 0.1
            elif key == 'p':
                v_angular -= 0.1
            elif key == 'k':
                v_linear_x = 0
                v_linear_y = 0
                v_angular = 0

            print(vel_info(v_linear_x, v_linear_y, v_angular))

            twist = Twist()
            twist.linear.x = v_linear_x
            twist.linear.y = v_linear_y
            twist.linear.z = 0
            twist.angular.x = 0
            twist.angular.y = 0
            twist.angular.z = v_angular

            publisher.publish(twist)

    except Exception as exception:
        print(exception)

    # set all parameters to 0 when the program is finished
    finally:
        twist = Twist()
        twist.linear.x = 0;  twist.linear.y = 0;  twist.linear.z = 0
        twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
        publisher.publish(twist)
        print("finish program")

    # finish program
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
```

暂时存在的问题有：

- 代码没有设置退出
- 并不知道实际的速度
- 需要测量最大速度
- 需要测量最小控制量，一次调节最小调节多少，和最大控制量
- 需要知道最短控制周期（也可以考虑写延迟函数以满足更长的控制周期）