ROS code

fgpk20

December 2021

1 Introduction

```
#!/usr/bin/env python

import rospy
import sys
import math
from nav_msgs.msg import Odometry
from sensor_msgs.msg import Imu,Range, LaserScan
from geometry_msgs.msg import Point, Twist
from rosgraph_msgs.msg import Clock
from math import atan2
from std_msgs.msg import Int32, Bool
from tf.transformations import euler_from_quaternion

#defining variables
Lrange = 10
Rrange = 10
absRDist=0.0
absLDist=0.0
maxLrange=0.0
maxRrange=0.0
angleToNormal=0.0
LeftLIDARlist=[]
RightLIDARlist=[]
incrimentAngle=0.0
currentTime=0
xVel=0.0
yVel=0.0
odDist=0.0
x_new=0.0
y_new=0.0
x_old=0.0
y_old=0.0

#proccesing functions
def leftRange(msg):
  global Lrange
  global absLDist
  global maxLrange
  Lrange=msg.range
  maxLrange=msg.max_range
  absLDist=Lrange/maxLrange
```

```python

def rightRange(msg):
  global Rrange
  global maxRrange
  global absRDist
  Rrange=msg.range
  maxRrange=msg.max_range
  absRDist=Rrange/maxRrange

def procImu(msg):
  global angleToNormal
  rot_q=msg.orientation
  (roll,pitch,angleToNormal)=euler_from_quaternion([rot_q.x, rot_q.
    y, rot_q.z, rot_q.w])
  global xVel
  global yVel
  xVel=msg.linear_acceleration.x
  yVel=msg.linear_acceleration.y

def LIDARprocess(msg):
  global LeftLIDARlist
  LeftLIDARlist = []
  global RightLIDARlist
  RightLIDARlist =[]
  global incrimentAngle
  incrimentAngle=msg.angle_increment
  LIDARangle=msg.angle_max
  for i in msg.ranges:
    if LIDARangle>0:
      LeftLIDARlist.append(i)
    else:
      RightLIDARlist.insert(0,i)
    LIDARangle-=msg.angle_increment
def SimTime(msg):
  global currentTime
  currentTime = msg.clock.secs

def newOdon(msg):
  global x_new
  global y_new
  x_new=msg.pose.pose.position.x
  y_new=msg.pose.pose.position.y


#subscriptuions
sub = rospy.Subscriber("/odom", Odometry,newOdon)
subImu = rospy.Subscriber("/imu", Imu,procImu)
subLeftIR = rospy.Subscriber("/range/fl", Range,leftRange)
subRightIR = rospy.Subscriber("/range/fr", Range,rightRange)
subLIDAR = rospy.Subscriber("/scan", LaserScan, LIDARprocess)
subClock=rospy.Subscriber("/clock",Clock,SimTime)
#publishing
pubMove = rospy.Publisher("/cmd_vel",Twist,queue_size=2)


rospy.init_node("single_project_node")
#"objectDetection"
```

```
#IR
def ObjectDetection(LIR,RIR):
  #determines if an object is ahead detected by IR
  if LIR<IRthreshold or RIR<IRthreshold:
    return True
  else:
    return False
#LIDAR

def LIDARdetection(List):
  #determines if an object is ahead detected by LIDAR
  angleOfLIDAR=0
  global LIDARthreshold
  maxRangeLIDAR=LIDARthreshold
  global incrimentAngle
  cWidthMin=0.2
  for j, i in enumerate(List):
    angleOfLIDAR=incrimentAngle*j
    if angleOfLIDAR <3.1415/8:
      cWidth=i*math.sin(angleOfLIDAR)
      if i <=maxRangeLIDAR and j in [0,1,2]:
        return True
      elif cWidth<cWidthMin and j!=0 and i<=maxRangeLIDAR:
        return True
    else:
      return False
def DistTravled(xSpeed,ySpeed,RosRate):
  time=float(1)/float(RosRate)
  xDist=xSpeed*time
  yDist=ySpeed*time
  Dist=(yDist**2+xDist**2)**0.5
  return Dist
def distTravledOd(x_new,x_old,y_new,y_old):
  x=x_old-x_new
  y=y_old-y_new
  dist=(y**2+x**2)**0.5
  return dist

#defined constants
RosRate=25
normalAngleAtOrigne=math.pi
LIDARthreshold=2.0
IRthreshold=0.7
#program
x_old=x_new
y_old=y_new
speed=Twist()

goal=Point()
flag=0
Count=2
rate = rospy.Rate(RosRate)
RateCount=0
LIDARflag=0
distTravledTotal=0.0
startTime=currentTime
distTravledOdom=0.0
```

```

while not rospy.is_shutdown():

  if Count!=0:
    Count-=1
    speed.linear.x=0.0
    speed.angular.z=0.0

  else:
    if ObjectDetection(Lrange,Rrange):
      if abs(Rrange-maxRrange)>0.1 and Lrange>=Rrange:

        speed.linear.x=0.1
        speed.angular.z=0.5
        flag=0

      elif abs(Lrange-maxLrange)>0.1 and Rrange>=Lrange:

        speed.linear.x=0.1
        speed.angular.z=-0.5
        flag=1

      else:
        speed.linear.x=0.05
        speed.angular.z=0.0
      RateCount=RosRate*4
      LIDARflag=0
    elif LIDARdetection(LeftLIDARlist):
      if LIDARflag==-1:
        speed.linear.x=0.05
        speed.angular.z=0.5
      else:
        speed.linear.x=0.05
        speed.angular.z=-0.5
        LIDARflag=1
    elif LIDARdetection(RightLIDARlist):
      if LIDARflag==1:
        speed.linear.x=0.05
        speed.angular.z=-0.5
      else:
        speed.linear.x=0.05
        speed.angular.z=0.5
        LIDARflag=-1

    else:
      #print 'object no longer detected'
      #print angleToNormal
      #gleToNormal<3.14159 and angleToNormal>0
      if RateCount!=0:
        speed.linear.x=0.2
        speed.angular.z=0.0
        RateCount-=1
      elif angleToNormal>0.05:
        speed.linear.x=0.2
        speed.angular.z=-0.25
        #print 'turning Left'
      elif angleToNormal<-0.05:
```

```
      speed.linear.x=0.2
      speed.angular.z=0.25
    else:
      speed.linear.x=0.2
      speed.angular.z=0.0
    LIDARflag=0
distTravledTotal+=DistTravled(speed.linear.x,0,RosRate)
distTravledOdom+=distTravledOd(x_new,x_old,y_new,y_old)
print 'time: ',currentTime-startTime,' dist: ', distTravledTotal, ' odom:',distTravledOdom
x_old=x_new
y_old=y_new
pubMove.publish(speed)
rate.sleep()
```