

# ROS code

fgpk20

December 2021

## 1 Introduction

```
1  #!/usr/bin/env python
2
3  import rospy
4  import sys
5  import math
6  from nav_msgs.msg import Odometry
7  from sensor_msgs.msg import Imu, Range, LaserScan
8  from geometry_msgs.msg import Point, Twist
9  from rosgraph_msgs.msg import Clock
10 from math import atan2
11 from std_msgs.msg import Int32, Bool
12 from tf.transformations import euler_from_quaternion
13
14 #defining variables
15 Lrange = 10
16 Rrange = 10
17 absRDist=0.0
18 absLDist=0.0
19 maxLrange=0.0
20 maxRrange=0.0
21 angleToNormal=0.0
22 LeftLIDARlist=[]
23 RightLIDARlist=[]
24 incrementAngle=0.0
25 currentTime=0
26 xVel=0.0
27 yVel=0.0
28 odDist=0.0
29 x_new=0.0
30 y_new=0.0
31 x_old=0.0
32 y_old=0.0
33
34 #processing functions
35 def leftRange(msg):
36     global Lrange
37     global absLDist
38     global maxLrange
39     Lrange=msg.range
40     maxLrange=msg.max_range
41     absLDist=Lrange/maxLrange
```

```

42
43 def rightRange(msg):
44     global Rrange
45     global maxRrange
46     global absRDist
47     Rrange=msg.range
48     maxRrange=msg.max_range
49     absRDist=Rrange/maxRrange
50
51 def procImu(msg):
52     global angleToNormal
53     rot_q=msg.orientation
54     (roll,pitch,angleToNormal)=euler_from_quaternion([rot_q.x, rot_q.
55         y, rot_q.z, rot_q.w])
56     global xVel
57     global yVel
58     xVel=msg.linear_acceleration.x
59     yVel=msg.linear_acceleration.y
60
61 def LIDARprocess(msg):
62     global LeftLIDARlist
63     LeftLIDARlist = []
64     global RightLIDARlist
65     RightLIDARlist =[]
66     global incrimentAngle
67     incrimentAngle=msg.angle_increment
68     LIDARAngle=msg.angle_max
69     for i in msg.ranges:
70         if LIDARAngle>0:
71             LeftLIDARlist.append(i)
72         else:
73             RightLIDARlist.insert(0,i)
74             LIDARAngle-=msg.angle_increment
75
76 def SimTime(msg):
77     global currentTime
78     currentTime = msg.clock.secs
79
80 def newOdon(msg):
81     global x_new
82     global y_new
83     x_new=msg.pose.pose.position.x
84     y_new=msg.pose.pose.position.y
85
86 #subscriptuions
87 sub = rospy.Subscriber("/odom", Odometry,newOdon)
88 subImu = rospy.Subscriber("/imu", Imu,procImu)
89 subLeftIR = rospy.Subscriber("/range/fl", Range,leftRange)
90 subRightIR = rospy.Subscriber("/range/fr", Range,rightRange)
91 subLIDAR = rospy.Subscriber("/scan", LaserScan, LIDARprocess)
92 subClock=rospy.Subscriber("/clock",Clock,SimTime)
93 #publishing
94 pubMove = rospy.Publisher("/cmd_vel",Twist,queue_size=2)
95
96 rospy.init_node("single_project_node")
97 #objectDetection"

```

```

98 #IR
99 def ObjectDetection(LIR,RIR):
100     #determines if an object is ahead detected by IR
101     if LIR<IRthreshold or RIR<IRthreshold:
102         return True
103     else:
104         return False
105 #LIDAR
106
107 def LIDARdetection(List):
108     #determines if an object is ahead detected by LIDAR
109     angleOfLIDAR=0
110     global LIDARthreshold
111     maxRangeLIDAR=LIDARthreshold
112     global incrementAngle
113     cWidthMin=0.2
114     for j, i in enumerate(List):
115         angleOfLIDAR=incrementAngle*j
116         if angleOfLIDAR <3.1415/8:
117             cWidth=i*math.sin(angleOfLIDAR)
118             if i <=maxRangeLIDAR and j in [0,1,2]:
119                 return True
120             elif cWidth<cWidthMin and j!=0 and i<=maxRangeLIDAR:
121                 return True
122         else:
123             return False
124 def DistTravled(xSpeed,ySpeed,RosRate):
125     time=float(1)/float(RosRate)
126     xDist=xSpeed*time
127     yDist=ySpeed*time
128     Dist=(yDist**2+xDist**2)**0.5
129     return Dist
130 def distTravledOd(x_new,x_old,y_new,y_old):
131     x=x_old-x_new
132     y=y_old-y_new
133     dist=(y**2+x**2)**0.5
134     return dist
135
136 #defined constants
137 RosRate=25
138 normalAngleAtOrigine=math.pi
139 LIDARthreshold=2.0
140 IRthreshold=0.7
141 #program
142 x_old=x_new
143 y_old=y_new
144 speed=Twist()
145
146 goal=Point()
147 flag=0
148 Count=2
149 rate = rospy.Rate(RosRate)
150 RateCount=0
151 LIDARflag=0
152 distTravledTotal=0.0
153 startTime=currentTime
154 distTravledOdom=0.0

```

```

155
156 while not rospy.is_shutdown():
157
158     if Count!=0:
159         Count-=1
160         speed.linear.x=0.0
161         speed.angular.z=0.0
162
163     else:
164         if ObjectDetection(Lrange,Rrange):
165             if abs(Rrange-maxRrange)>0.1 and Lrange>=Rrange:
166
167                 speed.linear.x=0.1
168                 speed.angular.z=0.5
169                 flag=0
170
171             elif abs(Lrange-maxLrange)>0.1 and Rrange>=Lrange:
172
173                 speed.linear.x=0.1
174                 speed.angular.z=-0.5
175                 flag=1
176
177             else:
178                 speed.linear.x=0.05
179                 speed.angular.z=0.0
180                 RateCount=RosRate*4
181                 LIDARflag=0
182         elif LIDARdetection(LeftLIDARlist):
183             if LIDARflag==-1:
184                 speed.linear.x=0.05
185                 speed.angular.z=0.5
186             else:
187                 speed.linear.x=0.05
188                 speed.angular.z=-0.5
189                 LIDARflag=1
190         elif LIDARdetection(RightLIDARlist):
191             if LIDARflag==1:
192                 speed.linear.x=0.05
193                 speed.angular.z=-0.5
194             else:
195                 speed.linear.x=0.05
196                 speed.angular.z=0.5
197                 LIDARflag=-1
198
199     else:
200         #print 'object no longer detected'
201         #print angleToNormal
202         #gleToNormal<3.14159 and angleToNormal>0
203         if RateCount!=0:
204             speed.linear.x=0.2
205             speed.angular.z=0.0
206             RateCount-=1
207         elif angleToNormal>0.05:
208             speed.linear.x=0.2
209             speed.angular.z=-0.25
210             #print 'turning Left'
211         elif angleToNormal<-0.05:

```

```

212         speed.linear.x=0.2
213         speed.angular.z=0.25
214     else:
215         speed.linear.x=0.2
216         speed.angular.z=0.0
217     LIDARflag=0
218     distTravledTotal+=DistTravled(speed.linear.x,0,RosRate)
219     distTravledOdom+=distTravledOd(x_new,x_old,y_new,y_old)
220     print 'time: ',currentTime-startTime,' dist: ', distTravledTotal,
        ' odom:',distTravledOdom
221     x_old=x_new
222     y_old=y_new
223     pubMove.publish(speed)
224     rate.sleep()

```