#!/usr/bin/env python

import rclpy

from rclpy.node import Node

from geometry\_msgs.msg import Twist

from sensor\_msgs.msg import LaserScan

import numpy as np

class ExplorerNode(Node):

def \_init\_(self):

super().\_init\_('explorer\_node')

self.laser\_sub = self.create\_subscription(

LaserScan, 'scan', self.laser\_callback, 10)

self.cmd\_vel\_pub = self.create\_publisher(Twist, 'cmd\_vel', 10)

self.obstacle\_threshold = 0.55 # Minimum safe distance to obstacles

self.move\_speed = 0.2 # Linear velocity for forward movement

self.rotation\_speed = 0.45 # Angular velocity for rotation (rad/s)

# PID controller parameters (can be fine-tuned for better behavior)

self.kp = 1.0 # Proportional gain

self.ki = 0.0 # Integral gain (currently disabled)

self.kd = 0.1 # Derivative gain

self.previous\_time = self.get\_clock().now()

self.prev\_error = 0.0 # Initialize previous error

def laser\_callback(self, msg: LaserScan):

cmd = Twist()

ranges = np.clip(np.array(msg.ranges), a\_min=0.18, a\_max=np.inf) # Clip invalid ranges

current\_time = self.get\_clock().now()

dt = (current\_time - self.previous\_time).nanoseconds / 1e9

self.previous\_time = current\_time

# Check for clear passage in front and back

if np.all(ranges[:20] > self.obstacle\_threshold) and np.all(ranges[-20:] > self.obstacle\_threshold):

# Clear path, move forward

cmd.linear.x = self.move\_speed

cmd.angular.z = 0.0

else:

# Obstacle detected, calculate error and adjust with PID control

min\_distance = min(np.min(ranges[:20]), np.min(ranges[-20:]))

error = self.obstacle\_threshold - min\_distance

angular\_z = self.kp \* error + self.kd \* (error - self.prev\_error) / dt

self.prev\_error = error # Update for next iteration

# Reduce linear speed while turning to avoid collisions

cmd.linear.x = self.move\_speed / 2.0

cmd.angular.z = np.clip(angular\_z, a\_min=-self.rotation\_speed, a\_max=self.rotation\_speed) # Limit rotation speed

self.cmd\_vel\_pub.publish(cmd)

def main(args=None):

rclpy.init(args=args)

explorer\_node = ExplorerNode()

rclpy.spin(explorer\_node)

rclpy.shutdown()

if \_name\_ == '\_main\_':

main()