CE-801-7-SP

INTELLIGENT SYSTEMS AND ROBOTICS

ASSIGNMENT REPORT

Submitted by,

Karan Bhatt

(**2112102**)

Submitted to,

Prof. Huosheng Hu

ABSTRACT

There are plenty of educators who have been trying to produce and automate the human reasoning and decision making. They ended up developing so many algorithms e.g., PID algorithm, Behaviour based control algorithm, Fuzzy Control etc. Here I am going to implement and presenting two different algorithms PID controller and Fuzzy logic on a mobile robot to execute the obstacles sense behaviour. First task in this project is consist of PID controller where robot can sense the obstacles through the laser, and it can move towards the target on given path by avoiding those obstacles. In Second task I am going to do the same but with the fuzzy logic where robot will perform same action as it does with PID controller but with Fuzzy logic. Later I will combine both the algorithms with the use of subsumption and context blending techniques. In real life we are using so many great inventions every day and even use of fuzzy logic is increasing. Where knowingly or unknowingly in our day-to-day life we are getting affected by it e.g., Self-driving cars, Robot Waiters/Chefs, AI based robot assistant etc.

Table of Content:

INTRODUCTION

The Robotics is a branch of technology that is consist of and deals with the construction, designing, operation of machines(robots) to perform various task. Machine can work in hazardous conditions where human cannot work. So, Robots have made it easy for human being to perform any task in such extreme environment. Wide use of robot is in automobile manufacture side to execute simple but repetitive tasks. Many aspects of robotics are consisted of artificial intelligence where they're inspired to study from human behaviours and animal in nature. Features Like, senses of touch, vision, temperature, behaviour, decision making, self-sufficiency. The term robotics was firstly introduced by the Isaac Asimov. He also quoted Three fundamental rules for robotics. Those three laws are from the "Handbook of Robotics, 56th Edition, 2058 A.D."

Rule 1: A robot may not injure a human being or, through inaction, allow a human being to come in harm.

Rule 2: A robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law.

Rule 3: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law. "

The field of robotics is growing emulously since last century here are the milestones that has been achieved in this field.

The History of Robots:

θ 1920 – Karel Capek used "Robot" in the play "Rossum's Universal Robots".

θ 1942 – Isaac Asimov published "Run-around" in Astounding Science Fiction, the "Three Laws of Robotics" was defined: later in book "I, Robot" (1950).

θ 1946 – J. Presper Eckert and John Mauchly built the ENIAC at the University of Pennsylvania - the 1st electronic computer.

θ 1951 –The first tele-operated articulated arm was developed for the Atomic Energy Commission by Raymond Goertz in France.

θ 1953 – A tortoise mobile robot was built by Grey Water in Bristol, U.K. θ 1959 – Marvin Minsky & John McCarthy established the AI Lab at MIT.

θ 1962 – Unimate manipulators were used on a production line at GM.

θ 1963 – John McCarthy led the new AI Laboratory at Stanford University.

θ 1965 – The Robotics Institute was established by Carnegie Mellon University.

θ 1968 – Kawasaki produced Unimation hydraulic robots in Japan.

θ 1970 – The Standard Arm was designed by Prof. Victor Scheinman.

θ 1978 – Using technology from Vicarm, Unimation develops the PUMA (Programmable Universal Machine for Assembly).

θ 1986 – Prof. R. Brooks at MIT: the research on behaviour-based robotics.

θ 1997 – NASA's Mars PathFinder mission on Mars (Sojourner rover robot). 2000 – Honda Asimo, the new generation of its series of humanoid robots.

θ 2003 – Sony Qiao humanoid robots.

θ 2005 – DARPA Grand Challenges (2004 and 2005); Urban Challenge (2007).

θ 2009 – Network Challenge (2009); Robotics Challenge Trials (2013).

θ 2015 – Robotics Challenge final (2015); Cyber Grand Challenge (2016).

θ 2017 – Subterranean Challenge (2017 -); Launch Challenge (2018 – present).

(9) (10)

For the programming in robotics most popular programming is Most probably C/C++. Python is also getting popular because of it is used in machine learning and even python can be used to develop ROS (Robotics Operating Systems) packages.

Robot Development platforms are intended to develop the robotics programs and make is feasible to use the robotic devices more intuitive. (9, 10)

Some of the popular Robot Development Platforms are stated below. (12)

1 Google ROBEL.

2 Microsoft AirSim.

3 Apollo Baidu.

4 NVIDIA Isaac.

5 AWS RoboMaker.

6 ROSbot 2.0.

7 Gazeebo

8 Poppy Project

Mobile Robot

Mobile robot is a machine which is controlled by software that uses sensors and controllers to identify its surroundings and work according to the environment. Mobile robots are autonomous systems that consist of so many components and technologies like sensor systems, Computer Systems, Actuator Systems, Mechanical Systems. And science like Physics Science, Artificial Intelligence, Control Theory. We can classify mobile robots in three different types of Legs (human like or animal like legs), tracks, wheels.

Mobile Robot

There are two types of mobile robots:

1) Autonomous mobile robots.

2) Non-Autonomous / Guided mobile robots.

Features of mobile robot:

- Wireless Communication
- Integrated Safety
- Fleet Simulation Software
- fleet management software
- integration with the company's supervisory software

Uses of mobile robots:

- shoreline exploration of mines.
- repairing ships.
- a robotic pack dog or exoskeleton to carry heavy loads for military troopers.
- robotic arms to assist doctors in surgery.
- painting and stripping machines or other structures.
- manufacturing automated prosthetics that imitate the body's natural functions.
- patrolling and monitoring applications, such as surveillant thermal and other environmental conditions.

Categories of Mobile robots:

- Stationary (arm/manipulator).
- Land-based. Wheeled mobile robot (WMR) Walking (or legged) mobile robot.
- Tracked slip/skid locomotion.
- Hybrid.
- Air-based.
- Water-based.

Sensors:

Robots are mainly working on their programmed sensors.

There are main two types of sensors

1. Internal Sensor
2. External Sensor

Internal Sensor

Internal Sensors establish its configuration in its own set of coordinate axes. Internal sensors are used to track the movement of the robot. How fast is he moving and at what angle is he looking? To determine the speed of movement of a robot, it is first necessary to determine the speed determined by the change in position and the direction of movement.

In internal sensors as well, there are three types:

1. Internal Position Sensor
2. Internal State Sensor
3. Internal Measurement Units

Internal Position Sensors are as below

- Contact sensors
- Micro switch -- to measure the danger situation (on/off output)
- Potentiometers -- to measure position (continuous output)
- Strain gauges -- to measure force or position (continuous output) non-contact sensors
- Optical interrupters -- to measure positions (on/off output)
- Incremental optical encoders -- to measure position & velocity (digital output)
- Absolute optical encoders -- to measure position (digital output)
- Synchro's -- to measure angular displacement (continuous output)
- Resolvers -- A rotary electrical transformer for measuring degrees of rotation.

External Sensors

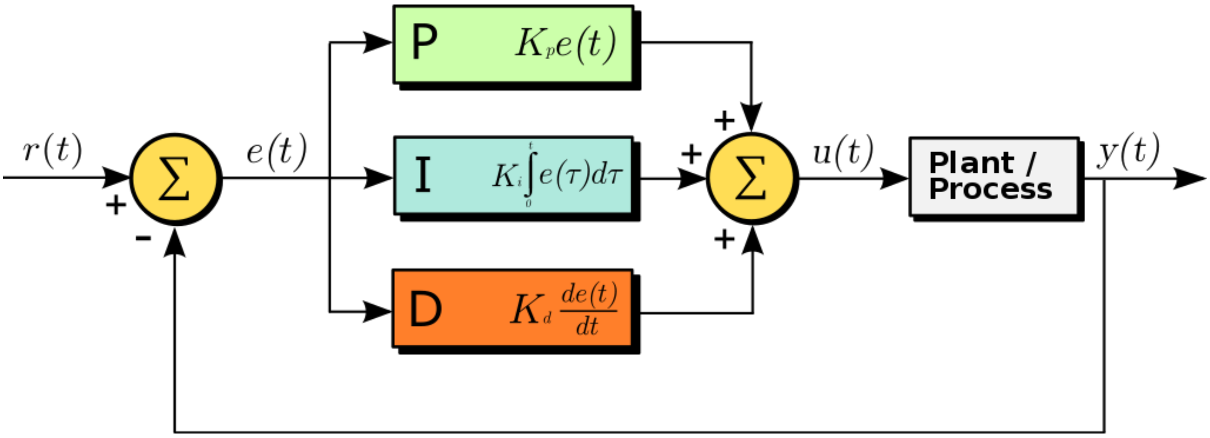
External sensors let the machine position itself to its environment. They are the device that senses the information of a control system, but They're not part of the systems. External Sensors locate the robot with respect to environment to travel around and complete a specific task. They're used to locate objects/obstacles in the environment with respect to the robot in order to handle them and avoid the collision. Provide real data to update the path of robotic system for path planning.

There are several types of External Sensors, few of them are stated below:

1. Tactile Sensors
2. Proximity Sensors
3. Range Finding Sensors
4. Triangulation Technique
5. Computer Visions
    a. Image Transformation
    b. Image Segmentation and analysis
    c. Image Understanding
    d. Real-world applications

PID Controller

Proportional Integral Derivative (PID) control algorithm is used to make close loop mechanism to control different systems, in particular case. This algorithm helps to maintain and complete the process at target level or value that we are working on. In this assignment, the task is to implement an obstacle avoidance and path finding behaviour on a robot with the help of PID controller. Here, we take the errors of desired and current location as an input of the PID controller. The output of this program is the proportional angular speed of the robot wheels.



$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\,\mathrm{d}\tau + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t},$$

This is the main formula of PID controller.

To Run the PID controller, we need to do some calculations of errors, as stated below in sequence:

e = desired distance - current distance

$e_i = e_i + e$

$e_d = e - e_{previous}$

$$e_{previous} = e$$

e, e_i and e_d are PID errors and k_p,k_i and k_d are PID parameters.
The PID output is then calculated as:

$$Output = k_p * e + k_i * e_i + k_d * e_d$$

To convert this output into left and right motor speeds of the robot:

$$leftVel = baseVel – (wd/2)$$

$$rightVel = baseVel + (wd/2)$$

Where baseVel is a constant, d is a constant distance between two wheels, w is a desired speed.

By running our code I got these graph data generated in Libreoffice Calc

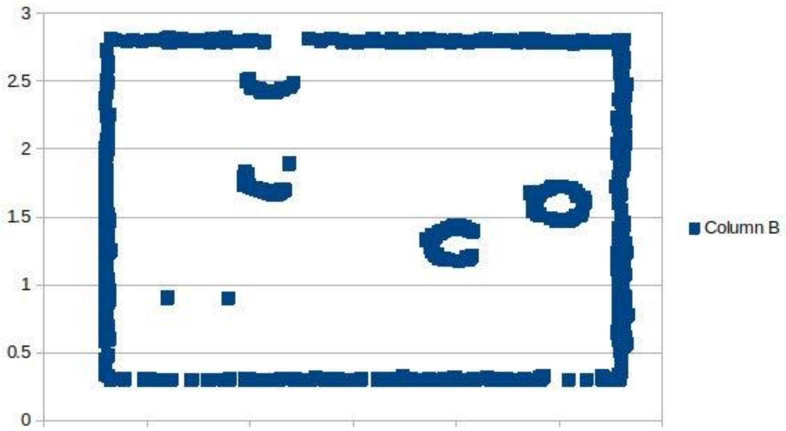


Figure 1: lasermap data

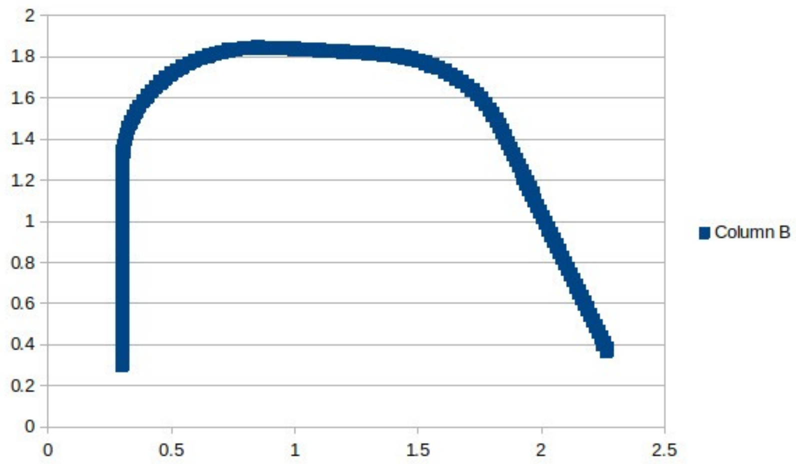


Figure 2: OdomTraj Data

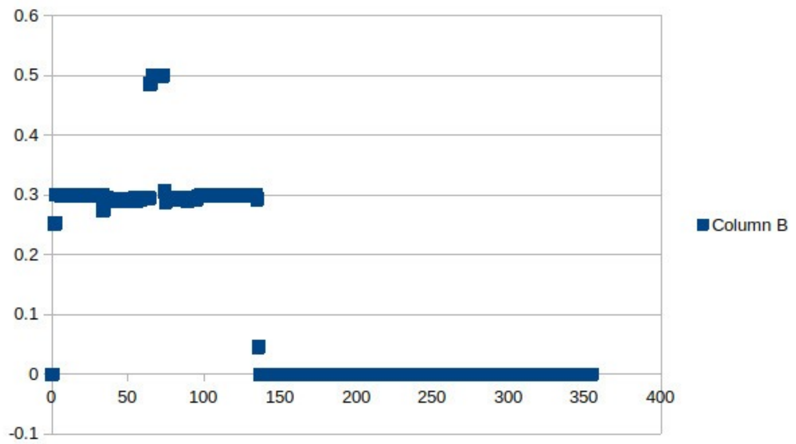
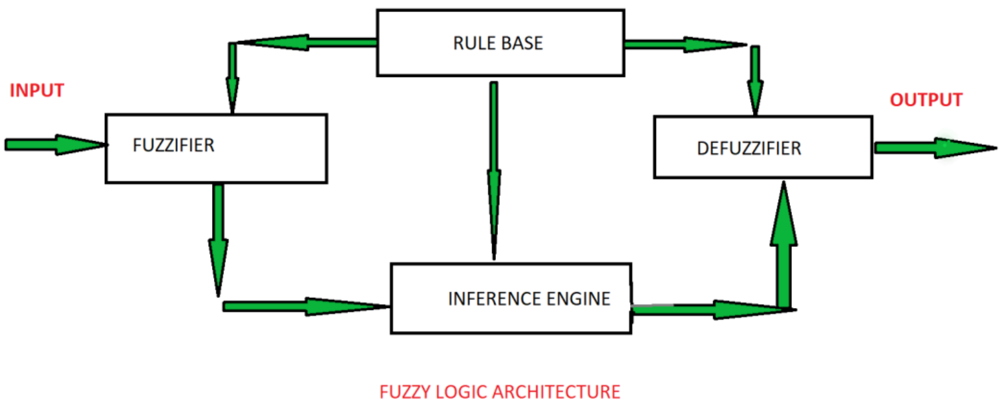


Figure 3: OdomVel Data

Fuzzy Logic

Fuzzy logic control is based on behaviour-based control of the robot. Behaviours can be any of action that comes to the category of directional behaviour/exploration, aversive/protective behaviours, goal-oriented behaviours. In complex, nonlinear, or indefinite systems for which there is strong practical information, fuzzy logic controllers frequently outperform other controllers. Fuzzy logic controllers are based on fuzzy sets, or classes of objects with a smooth rather than abrupt transition from membership to no membership. In this assignment I tried to perform a task of obstacles avoidance for a robot using fuzzy logic control.



This figure is describing the process of fuzzy logic control.

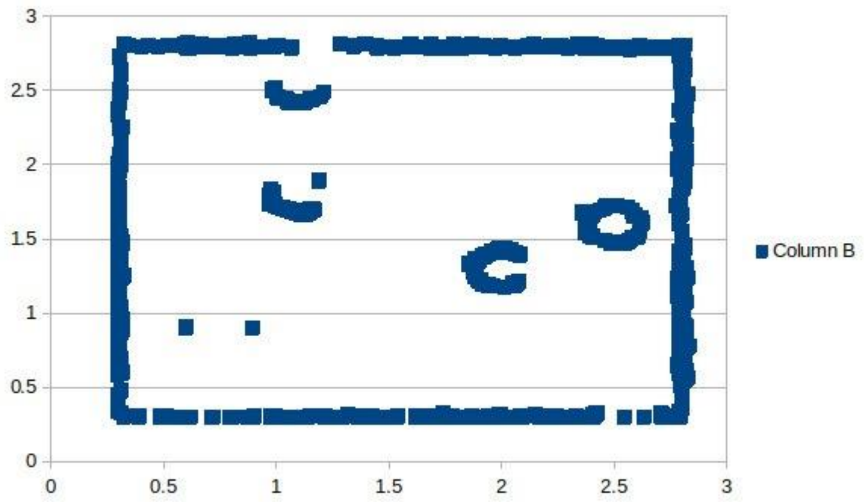
Obstacle Avoidance:

We have inputs in obstacle avoidance behaviour of The sensors. The outputs, particularly the robot's left motor speed and right motor speed, stay unchanged. In this scenario, the rule base is attempting to develop a protective behaviour for the robot. As a result, if it gets too close to a right-hand object, it moves left. Similarly, if the robot gets too close to a left-hand object, it turns right. If it senses a very close object in front of it, it can turn left or right.

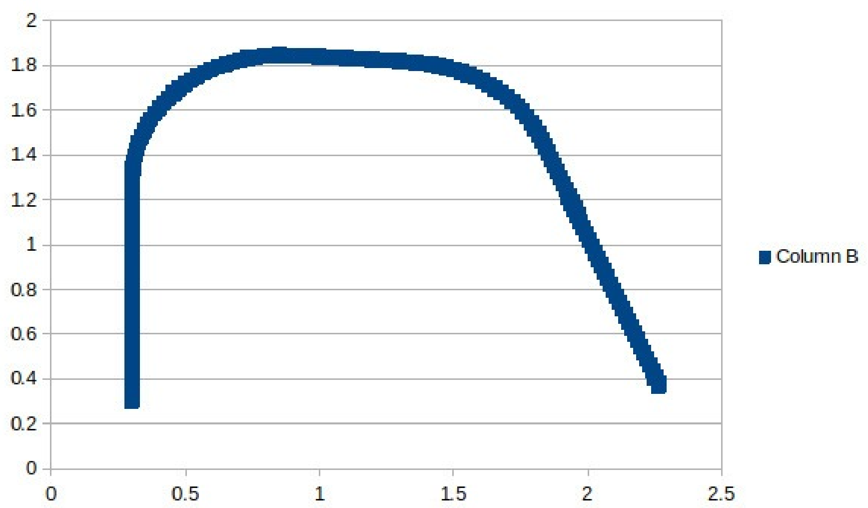
Application of fuzzy logic

Fuzzy logic is being used into different different systems which helps us in our daily life. From a small vacuum cleaner to smart robots with human intelligence, everywhere we can use fuzzy logic. Some of the automation things like fire detectors also consist of it which tells us by sensing if there is a fire. In the self-driving cars as well, it detects the objects on the road and even sides of the road and try not to crash with anything. Fuzzy logic is used to make data analysis and data mining simple to some point. Researchers are studying/improving on the giving power to the matching that can take its own decision by its own just as human being with the help of fuzzy logic. Hence, the use of fuzzy logic is increasing day by day.

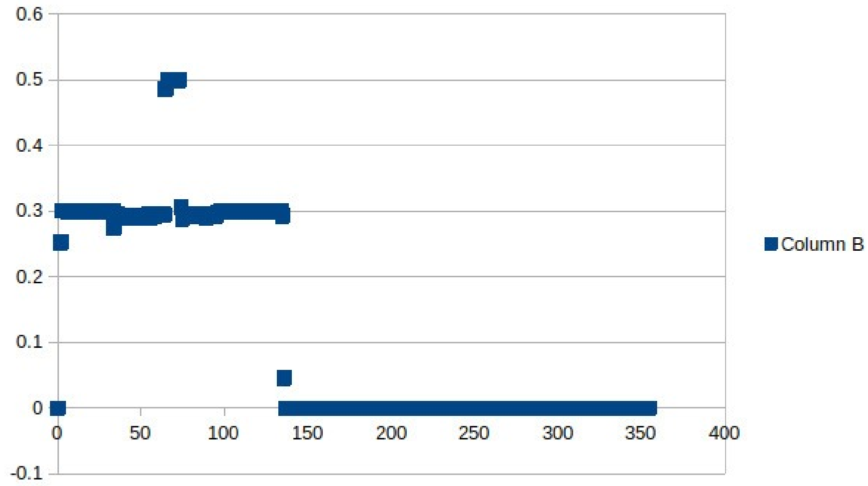
By running our code, I got these graph data generated in LibreOffice Calc



LaserMap Data



OdomTraj Data



OdomVel Data

References

1) https://www.britannica.com/technology/robotics
2) Debasish Pal and Debasish Bhattacharya.
3) human work efficiency in government offices, private organizations, and commercial
4) business centres in agartala city using fuzzy expert system: A case study. Advances
5) in Fuzzy Systems, 2012
6) S.L Anderson. Asimov's "three laws of robotics" and machine metaethics. AI
7) Soc 22, 477–493, 2008.
8) https://kambria.io/blog/tech/robotics
9) https://www.amnh.org/explore/news-blogs/news-posts/revisiting-asimovs-three-laws-of-robotics
10) https://www.aventine.org/robotics/history-of-robotics
11) https://www.techtarget.com/whatis/definition/robotics
12) https://ohmnilabs.com/content/robot-development-platforms-technologies-and-trends/
13) https://www.techtarget.com/iotagenda/definition/mobile-robot-mobile-robotics
14) https://www.techtarget.com/iotagenda/definition/mobile-robot-mobile-robotics
15) https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/
16) https://sites.google.com/site/robotsitething/history
17) https://journals.sagepub.com/doi/full/10.1177/1729881419839596
18) https://en.wikipedia.org/wiki/PID_controller
19) https://www.techtarget.com/searchenterpriseai/definition/fuzzy-logic#:~:text=Fuzzy%20logic%20is%20an%20approach,at%20Berkeley%20in%20the%201960s.
20) https://en.wikipedia.org/wiki/Fuzzy_logic
21) https://www.geeksforgeeks.org/fuzzy-logic-introduction/
22) https://www.omega.co.uk/prodinfo/pid-controllers.html#:~:text=A%20PID%20controller%20is%20an,most%20accurate%20and%20stable%20controller.

Code: C/C++

This code is consisted of PID controller as well as Fuzzy logic controller

```
#include <chrono>
#include <functional>
#include <memory>
#include <string>
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
#include "geometry_msgs/msg/twist.hpp"
#include "sensor_msgs/msg/laser_scan.hpp"
#include "geometry_msgs/msg/pose.hpp"
#include "nav_msgs/msg/odometry.hpp"

#include <fstream>
#include <time.h>
#include <iomanip>

struct EulerAngles{double roll, pitch, yaw;}; // yaw is what you want, i.e. Th
struct Quaternion{double w, x, y, z;};

struct PID_para{double kp, ki, kd, ei_pre, ed_pre, Max_output;};

double PID_control(PID_para pid, double setPoint, double measuredData)
```

```
{
double err = setPoint - measuredData;
double ei = pid.ei_pre + err;
double ed = err - pid.ed_pre;
double output = pid.kp*err + pid.ki*ei + pid.kd*ed;
if (output > pid.Max_output)
output = pid.Max_output;
else if(output < -pid.Max_output)
output = -pid.Max_output;
pid.ei_pre = ei;
pid.ed_pre = ed;
return output;
}

EulerAngles ToEulerAngles(Quaternion q){ // for calculating Th
EulerAngles angles;
// roll (x-axis rotation)
double sinr_cosp = +2.0 * (q.w * q.x + q.y * q.z);
double cosr_cosp = +1.0 - 2.0 * (q.x * q.x + q.y * q.y);
angles.roll = atan2(sinr_cosp, cosr_cosp);
// pitch (y-axis rotation)
double sinp = +2.0 * (q.w * q.y - q.z * q.x);
if (fabs(sinp) >= 1)
angles.pitch = copysign(M_PI/2, sinp); //use 90 degrees if out of range
else
angles.pitch = asin(sinp);
```

```
// yaw (z-axis rotation)
double siny_cosp = +2.0 * (q.w * q.z + q.x * q.y);
double cosy_cosp = +1.0 - 2.0 * (q.y * q.y + q.z * q.z);
angles.yaw = atan2(siny_cosp, cosy_cosp);
return angles;
}

using namespace std::chrono_literals;

using namespace std;

ofstream laserFile; // Declare a file object for recording your laser data.
ofstream laserMapFile;
ofstream odomTrajFile;
class Stopper : public rclcpp::Node{
public:
/* velocity control variables*/
constexpr const static double FORWARD_SPEED_LOW = 0.1;
constexpr const static double FORWARD_SPEED_MIDDLE = 0.3;
constexpr const static double FORWARD_SPEED_HIGH = 0.5;
constexpr const static double FORWARD_SPEED_STOP = 0;
constexpr const static double TURN_LEFT_SPEED_LOW = 0.3;
constexpr const static double TURN_LEFT_SPEED_MIDDLE = 0.6;
constexpr const static double TURN_LEFT_SPEED_HIGH = 1.0;
constexpr const static double TURN_RIGHT_SPEED_LOW = -0.3;
constexpr const static double TURN_RIGHT_SPEED_MIDDLE = -0.6;
```

```
constexpr const static double TURN_RIGHT_SPEED_HIGH = -1.0;

constexpr const static double TURN_SPEED_ZERO = 0;

/* class constructor */

Stopper():Node("Stopper"), count_(0){

publisher_=this->create_publisher<geometry_msgs::msg::Twist>("cmd_vel",
10);

odomSub_=this-
>create_subscription<nav_msgs::msg::Odometry>("odom",10,std::bind(&Stop
per::odomCallback, this, std::placeholders::_1));


laserScan_=this-
>create_subscription<sensor_msgs::msg::LaserScan>("scan",10,

std::bind(&Stopper::scanCallback, this, std::placeholders::_1));


};

/* moving function */

void startMoving();

void moveStop();

void moveForward(double forwardSpeed);

void moveRight(double turn_right_speed);

void moveForwardRight(double forwardSpeed, double turn_right_speed);

void odomCallback(const nav_msgs::msg::Odometry::SharedPtr odomMsg);


double PositionX=0.3, PositionY=0.3, homeX=0.3, homeY=0.3;

double odom_landmark1=1.20, odom_landmark1a=0.38,
odom_landmark2=0.80;

double laser_landmark1 = 1.3, laser_landmark2 = 1.4;

double laser_landmark3 =0.53, laser_landmark4 = 0.7, laser_landmark5 = 0.3;
```

```
int stage=1;
double odom_landmark3=1.20, odom_landmark4=1.80,
odom_landmark5=2.25;

void scanCallback(const sensor_msgs::msg::LaserScan::SharedPtr scan);
double frontRange, mleftRange, leftRange, rightRange, mrightRange;
int laser_index = 0; // index the laser scan data

Quaternion robotQuat;
EulerAngles robotAngles;
double robotHeadAngle;
double leftAngle = M_PI/2, mleftAngle = M_PI/4, frontAngle=0;
double mrightAngle = -M_PI/4, rightAngle = -M_PI/2;
void transformMapPoint(ofstream& fp, double laserRange, double
laserTh,double robotTh, double robotX, double robotY);

void PID_wallFollowing(double forwardSpeed, double laserData);
void PID_pass1stGap(double moveSpeed, double robotHeading);

void Fuzzy_wallFollowing(double laserData1, double laserData2);
void Fuzzy_to1stGap(double laserData1, double laserData2);

private:
// Publisher to the robot's velocity command topic
rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr publisher_;
rclcpp::TimerBase::SharedPtr timer_;
size_t count_;
```

```
//Subscriber to robot's odometry topic
rclcpp::Subscription<nav_msgs::msg::Odometry>::SharedPtr odomSub_;
rclcpp::Subscription<sensor_msgs::msg::LaserScan>::SharedPtr laserScan_;

};
void Stopper::moveStop(){
auto msg = geometry_msgs::msg::Twist();
msg.linear.x = FORWARD_SPEED_STOP;
publisher_->publish(msg);
}
void Stopper::moveForward(double forwardSpeed){
//The default constructor to set all commands to 0
auto msg=geometry_msgs::msg::Twist();
//Drive forward at a given speed along the x-axis.laser_landmark2
msg.linear.x = forwardSpeed;
publisher_->publish(msg);
}
void Stopper::moveRight(double turn_right_speed){
auto msg = geometry_msgs::msg::Twist();
msg.angular.z = turn_right_speed;
publisher_->publish(msg);
}

void Stopper::moveForwardRight(double forwardSpeed, double
turn_right_speed){
auto msg = geometry_msgs::msg::Twist();
msg.linear.x = forwardSpeed;
```

```
msg.angular.z = turn_right_speed;

publisher_->publish(msg);

}

void Stopper::odomCallback(const nav_msgs::msg::Odometry::SharedPtr
odomMsg){

PositionX = odomMsg->pose.pose.position.x + homeX;

PositionY = odomMsg->pose.pose.position.y + homeY;

RCLCPP_INFO(this->get_logger(),"RobotPostion: %.2f , %.2f",PositionX,
PositionY );

RCLCPP_INFO(this->get_logger(), "Robot stage: %d ", stage );

/* if (PositionY < odom_landmark1 && PositionX < odom_landmark1a){

stage = 1;

moveForward(FORWARD_SPEED_MIDDLE);

}

else if (PositionX < odom_landmark2){

stage =2;

moveForwardRight(FORWARD_SPEED_MIDDLE,TURN_RIGHT_SPEED_MIDDLE);

}

else if (PositionX < odom_landmark3){

stage = 3;

moveForward(FORWARD_SPEED_HIGH);

}mrightRange

else if (PositionX < odom_landmark4){

stage = 4;laser_landmark4

moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);

}
```

```
else if (PositionX < odom_landmark5){
stage = 5;laser_landmark3
moveForward(FORWARD_SPEED_MIDDLE);laser_landmark5
}
else{
stage = 6;
moveStop();
} */
odomTrajFile<< PositionX <<" "<< PositionY<<endl;
robotQuat.x = odomMsg->pose.pose.orientation.x;
robotQuat.y = odomMsg->pose.pose.orientation.y;
robotQuat.z = odomMsg->pose.pose.orientation.z;
robotQuat.w = odomMsg->pose.pose.orientation.w;
robotAngles = ToEulerAngles(robotQuat);
robotHeadAngle = robotAngles.yaw;
}

void Stopper::PID_wallFollowing(double forwardSpeed, double laserData)
{
PID_para controller;
double landmark1_toWall = 0.3;
controller.kp = 0.1, controller.ki = 0.01;
controller.kd = 0.001, controller.Max_output = 0.6;
double PID_output = PID_control(controller, landmark1_toWall, laserData);
moveForwardRight(forwardSpeed, PID_output);
}
```

```
void Stopper::PID_pass1stGap(double moveSpeed, double robotHeading)
{
PID_para controller;
double robotHeadingGap1 = 0; // the robot heading should be 0 degree
controller.kp = 2, controller.ki = 0.01;
controller.kd = 0.001, controller.ei_pre = 0;
controller.ed_pre = 0, controller.Max_output = 0.6;
double PID_output = PID_control(controller, robotHeadingGap1,
robotHeading);
moveForwardRight(moveSpeed, PID_output);
}

void Stopper::Fuzzy_wallFollowing(double laserData1, double laserData2)
{
int fuzzySensor1, fuzzySensor2;
// sensor data fuzzification
if (laserData1 < 0.3) fuzzySensor1 = 1; // The robot is near to the wall
else if (laserData1 < 0.5) fuzzySensor1 = 2; // The robot is on the right distance
else fuzzySensor1 = 3; // The robot is far from the wall;
if (laserData2 < 0.4) fuzzySensor2 = 1; // The robot is near to the wall
else if (laserData2 < 0.6) fuzzySensor2 = 2; // The robot at the right distance;
else fuzzySensor2 = 3; // The robot is far from the wall;
// Fuzzy rule base and control output
if (fuzzySensor1 == 1 && fuzzySensor2 == 1)
moveForwardRight(FORWARD_SPEED_LOW, TURN_RIGHT_SPEED_LOW);
else if (fuzzySensor1 == 1 && fuzzySensor2 == 2)
```

```
moveForwardRight(FORWARD_SPEED_LOW, TURN_RIGHT_SPEED_LOW);
else if (fuzzySensor1 == 1 && fuzzySensor2 == 3)
moveForwardRight(FORWARD_SPEED_LOW, TURN_LEFT_SPEED_LOW);
else if (fuzzySensor1 == 2 && fuzzySensor2 == 1)
moveForwardRight(FORWARD_SPEED_MIDDLE, TURN_RIGHT_SPEED_LOW);
else if (fuzzySensor1 == 2 && fuzzySensor2 == 2)
moveForwardRight(FORWARD_SPEED_HIGH, TURN_SPEED_ZERO);
else if (fuzzySensor1 == 2 && fuzzySensor2 == 3)
moveForwardRight(FORWARD_SPEED_MIDDLE, TURN_LEFT_SPEED_LOW);
else if (fuzzySensor1 == 3 && fuzzySensor2 == 1)
moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);
else if (fuzzySensor1 == 3 && fuzzySensor2 == 2)
moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);
else if (fuzzySensor1 == 3 && fuzzySensor2 == 3)
moveForwardRight(FORWARD_SPEED_HIGH, TURN_LEFT_SPEED_LOW);
else RCLCPP_INFO(this->get_logger(), “Following the left wall”);
}

void Stopper::Fuzzy_to1stGap(double laserData1, double laserData2)
{
int fuzzySensor1, fuzzySensor2;
// sensor data fuzzification
if (laserData1 < 0.4) fuzzySensor1 = 1;
else if (laserData1 < 0.6) fuzzySensor1 = 2;
else fuzzySensor1 = 3;
```

```
if (laserData2 < 0.4) fuzzySensor2 = 1;

else if (laserData2 < 0.8) fuzzySensor2 = 2;

else fuzzySensor2 = 3;

// Fuzzy rule base and control output

If (fuzzySensor1 == 1 && fuzzySensor2 == 1)

moveForwardRight(FORWARD_SPEED_LOW, TURN_RIGHT_SPEED_LOW);

else if (fuzzySensor1 == 1 && fuzzySensor2 == 2)

moveForwardRight(FORWARD_SPEED_LOW, TURN_RIGHT_SPEED_LOW);

else if (fuzzySensor1 == 1 && fuzzySensor2 == 3)

moveForwardRight(FORWARD_SPEED_LOW, TURN_LEFT_SPEED_LOW);

else if (fuzzySensor1 == 2 && fuzzySensor2 == 1)

moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);

else if (fuzzySensor1 == 2 && fuzzySensor2 == 2)

moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);

else if (fuzzySensor1 == 2 && fuzzySensor2 == 3)

moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);

else if (fuzzySensor1 == 3 && fuzzySensor2 == 1)

moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);

else if (fuzzySensor1 == 3 && fuzzySensor2 == 2)

moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);

else if (fuzzySensor1 == 3 && fuzzySensor2 == 3)

moveForwardRight(FORWARD_SPEED_HIGH, TURN_RIGHT_SPEED_MIDDLE);

else RCLCPP_INFO(this->get_logger(), “Going through the 1st gap”);
```

```
}

void Stopper::scanCallback(const sensor_msgs::msg::LaserScan::SharedPtr scan)
{
leftRange = scan->ranges[300]; // get a range reading at the left angle
mleftRange = scan->ranges[250]; // get a range reading at the front-left angle
frontRange = scan->ranges[200]; // get a range reading at the front angle
mrightRange = scan->ranges[150]; // get a range reading at the front-right angle
rightRange = scan->ranges[100]; // get the range reading at the right angle
laserFile << leftRange << ","<< mleftRange << "," << frontRange<<"," <<
mrightRange << "," << rightRange <<"," <<laser_index++<< endl;
transformMapPoint(laserMapFile,frontRange,frontAngle,robotHeadAngle,PositionX, PositionY);
transformMapPoint(laserMapFile, mleftRange, mleftAngle, robotHeadAngle,
PositionX, PositionY);
transformMapPoint(laserMapFile, leftRange, leftAngle, robotHeadAngle,
PositionX, PositionY);
transformMapPoint(laserMapFile, rightRange, rightAngle, robotHeadAngle,
PositionX, PositionY);
transformMapPoint(laserMapFile, mrightRange, mrightAngle, robotHeadAngle,
PositionX, PositionY);

/**
switch(stage){
case 1:
```

```
if (frontRange > laser_landmark1)
moveForward(FORWARD_SPEED_MIDDLE);
else stage = 2;
break;
case 2:
if (mleftRange < laser_landmark2)
moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);
else stage = 3;
break;
case 3:
if (frontRange > laser_landmark3)
moveForward(FORWARD_SPEED_MIDDLE);
else stage = 4;
break;

case 4:
if (rightRange > 0.3 )
moveForwardRight(FORWARD_SPEED_MIDDLE,
TURN_RIGHT_SPEED_MIDDLE);
else stage = 5;
break;

case 5:
if ( frontRange > laser_landmark5)
moveForward(FORWARD_SPEED_MIDDLE);
else stage = 6;
```

```
break;
case 6:
moveStop();
break;
}
}
**/

switch(stage){
case 1: // wall following
if (PositionY < odom_landmark1 && PositionX < odom_landmark1a)
PID_wallFollowing(FORWARD_SPEED_MIDDLE, leftRange);
//Fuzzy_wallFollowing(leftRange, mleftRange);
else stage = 2;
break;
case 2: // going through the 1st gap
if (PositionX < odom_landmark2)
PID_pass1stGap(FORWARD_SPEED_MIDDLE, robotHeadAngle);
//Fuzzy_to1stGap(leftRange, mleftRange);
else stage = 3;
break;
case 3:
if (PositionX < odom_landmark3)
PID_pass1stGap(FORWARD_SPEED_MIDDLE, 45);
//Fuzzy_to1stGap(leftRange, mleftRange);
else stage = 4;
```

```
break;
case 4:
if (PositionX < odom_landmark4)
PID_pass1stGap(FORWARD_SPEED_MIDDLE, 0);
//Fuzzy_to1stGap(leftRange, mleftRange);
else stage = 5;
break;
case 5:
if ( PositionX < odom_landmark5)
PID_wallFollowing(FORWARD_SPEED_MIDDLE, 60);
//Fuzzy_to1stGap(leftRange, mleftRange);
else stage = 6;
break;
case 6:
moveStop();
break;
}
}

void Stopper::transformMapPoint(ofstream& fp, double laserRange, double
laserTh,
double robotTh, double robotX, double robotY)
{
double transX, transY;
transX = laserRange * cos(robotTh + laserTh) + robotX;
transY = laserRange * sin(robotTh + laserTh) + robotY;
if (transX < 0) transX = homeX; else transX += homeX;
```

```
if (transY < 0) transY = homeX; else transY += homeY;

fp << transX << ", " << transY << endl;

}

void Stopper::startMoving(){

odomTrajFile.open("/home/kb21030/M-Drive/ros_workspace/src/tutorial_pkg/odomTrajData.csv",ios::trunc); //note you should modify hhu to your username

odomVelFile.open("/home/kb21030/M-Drive/ros_workspace/src/tutorial_pkg/odomVelData.csv", ios::trunc);

laserFile.open("/home/kb21030/M-Drive/ros_workspace/src/tutorial_pkg/laserData.csv",ios::trunc);

laserMapFile.open("/home/kb21030/M-Drive/ros_workspace/src/tutorial_pkg/laserMapData.csv",ios::trunc);

RCLCPP_INFO(this->get_logger(), "Start moving");

rclcpp::WallRate loop_rate(10);

while (rclcpp::ok()){

auto node = std::make_shared<Stopper>();

rclcpp::spin(node); // update

loop_rate.sleep(); // wait delta time

}

odomTrajFile.close();

laserFile.close();

laserMapFile.close();

}


int main(int argc, char *argv[]){

rclcpp::init(argc, argv);

Stopper stopper;
```

```
stopper.startMoving();
return 0;
}
```