INTELLIGENT SYSTEMS AND ROBOTICS ASSIGNMENT REPORT

A REPORT

submitted as part of the assignment

In

CE801 - Intelligent systems and robotics

by

supasun khumpraphan
(sk21395)

School of Computer Science and Electronic Engineering
University of Essex
January 2022

Abstract

In an era where robots are increasingly playing a role in our daily lives, human beings strive to create robots that are self-adjusting and capable of making decisions that are the same or superior to humans but creating truly flexible and efficient robotics software is difficult because when using the robot, there are often various problems according to the environment or the place where it is used. Dealing with those problems is difficult. Robot development in the past was private development. causing knowledge or information to be kept in one person or kept secret in that organization This form of development makes it difficult to develop robots because developers need to develop robots themselves in every part, whether it is a robot's perception system, or robot propulsion system. Including various complex work processes, resulting in ROS being used to enable the rapid development of complex robots. ROS is built based on supporting the development of software for robots that can work together from multiple parts. ROS is specifically designed to work together. In this study, two types of surveys will be conducted, namely. Type 1 is a PID controller that controls actions and moves easily and it can not beyond expectation. This makes it inflexible enough to control complex or frequently stirred systems. however, Type 2 is fuzzy logic, it can think outside the box and control complex actions. And many things can be seen today such as Self-Driving cars and cleaning house robots.

.

Table Contents

1. Introduction

The past decade has been a time of innovation where everything was developed to meet human needs, especially inventing things to help people like robot innovation with a variety of technologies Robots have spread almost everywhere. Be it hospitals, universities, schools, or even our own homes and of course, the place where robot innovation is used the most is inevitably the industrial plant. Therefore, the study of robot operation is very necessary and important.

1.1.Type of robot

Robots can be divided into two types are the first type is a fixed robot, which is a robot that cannot move on its own. It looks like a mechanical arm that can move and move only each joint only within oneself Often used in industrial plants such as car assembly plants. The second type is a mobile robot. This is different from a fixed robot because they can move around by themselves and using wheels or using legs, which is this type of robot. At present, it is still a research study conducted in a laboratory to develop to use in various forms such as Mars exploration robots of NASA. Nowadays, have development of robots to look like pets like dogs to become friends with humans. For example, the Sony IBO robot or even the development of robots to be able to move like humans on two legs so that in the future it can be used in jobs that are risky for danger instead of humans.

Fig. 1: self-driving robots with Hermes company in London[1].

1.2.Robot movement

The movements of robots can be divided into six major categories. The first type, wheel-drive locomotion, is a robot that uses wheels to move. Suitable for general robots operating on level ground. The advantages are The robot will be able to move quickly. easy

control, Therefore, most robots are built as robots that move using wheels. The limitations of this type of motion are Robots can't go to different level areas. Traveling in rugged terrain can be difficult. The second type, track-drive locomotion, is a robot that uses belt wheels to move. Suitable for robots operating in rugged terrain. or areas with different level areas.
The controls can be as simple as a normal wheeled robot. The limitation is that the robot cannot move at high speed. and may cause damage to the surface area where the robot moves. from the scratch of the belt wheel. The third type, legged locomotion, is a robot that uses its legs to move. by imitating living things such as four-legged walking robots or a robot that walks on two legs Advantage of robots that use legs is that robots can go anywhere on all surface conditions able to step over various obstacles It has better maneuverability than wheels. The limitation is slow motion The controls are much more difficult than wheeled movement. And maintaining balance is very necessary for this type of robot. Especially robots that use two legs to move. The fourth type is Moving in water (swimming locomotion) is a robot that uses propellers or fins to move and there is a ballast tank to control the buoyancy of the robot. which are robot fish and submarine robots which are mostly used in survey work Caution of robots moving in water since underwater motion cannot use photos for navigation, the control must be guided by other material, such as a sound wave reflex system. The controls, therefore, have to be very careful. Type five is flight locomotion designed by bringing wings. or propellers to be used as a robotic propulsion system to float high above the ground It has a movement pattern similar to that of an airplane or helicopter. Suitable for aerial movement, survey work, and surveillance. Advantages The robot can move in all conditions such as risky areas. Areas that are difficult to reach on land or water. it has a high speed and it has a long operating distance. Limitations because the robot has a long operating distance. control system design or remote control must be highly efficient An hard control system can cause the robot to crash. the last type is Other locomotion is a robot that does not use legs and wheels to move, such as The snake robot uses a combination consolidation of the resultant forces generated by the twisting movements of each joint. drive to move forward The advantages of this type of robot are can go on any surface able to go up high or low and also can enter narrow spaces therefore able to perform various tasks. another advantage of this type of robot is that each joint of the assembled robot same will be the same So if some joints are damaged can be replaced immediately with another joint.

1.3.Type of Wheeled Mobile Robots

There are many different types of Wheeled Mobile Robots. The first type is Tricycle Drive Mobile Robots. The advantage is that it does not anti-vibration system required and can turn the wheels at all without moving forward[6]. The center of the pivot center is at the midpoint of the rear axle and must be controlled with the front wheels and moving with the rear wheels. The second type is Car-like Mobile Robots that use only walking or backward to turn, cannot turn immediately, and use outer wheels to steer. Type 3 is Differential Drive Mobile Robots can be driven independently. It has casters for stability and can effectively rotate around the center. The fourth type is Tracked Mobile Robots, used for surveying in rugged road

conditions but there is a terrible measure of distance because the skid has to be controlled instead. The last type is Multi-Degree-of-Freedom Vehicles that are extremely maneuverable and high maneuverability in tight spaces but must be controlled and coordinated with many motors

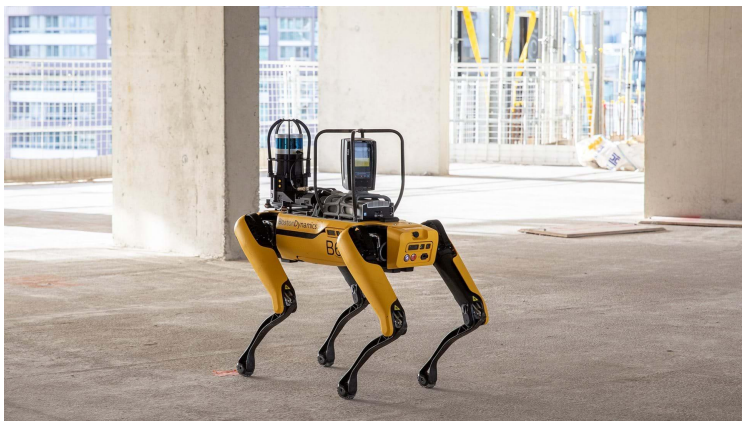

Fig. 2: A four-legged robot works on the Construction Site[2].

1.4. Type of Legged Walking Robots

Legged Walking Robots can be divided into two types. firstly, the dynamic type of the robot must maintain a good center of gravity because it has to have a good balance. After all, it only has one or two legs. Another type is Static stable systems to must be placed feet contact points are positioned properly because the center of gravity is at the feet contact points. One-Legged Walking Robot uses only one leg, it has to move only by jumping forward. While Two-legged Walking Robots have two human-like legs, they can do a lot of human-like things such as walking, running in rough terrain, and jumping over obstructions. And finally, Multi-legged Walking Robots can be controlled like One-Legged Walking robots and Two-legged Walking Robots. However, no matter how many legs they have, they are still more difficult to control than Wheeled Mobile Robots.

1.5. Sensors

To issue commands so that the robot can move automatically. The robot must have a location recognition part of both the robot itself and the destination to be used to calculate the appropriate route and able to avoid obstacles What will give the robot the ability to sense the environment is "Sensors consist of internal sensors that measure variables that have inside the robot such as acceleration, velocity, and positioning. place of the robot To travel from place to place with precision and External sensors that measure variables from outside the vehicle. to find the best place to travel during that time By avoiding obstacles from a collision. Various sensors help in many situations and can be done easily and various sensors help in detection: Proximity sensors that detect the surroundings by light or Range finding sensors that are used to avoid collisions, mapping, collecting environments such as trees and rivers Image transformation

involves the conversion of image features into digital signal format Image segmentation includes edge detection to use edge detection, zoning to bring each zone able to analyze[7].

2. PID Controller

The Proportional Integral Derivative (PID) is a Proportional, Integral, and Derivative control system. It is a widely used feedback control system. which the value used in the calculation is the error value is as low as possible by adjusting the input signal of the variable values process of PID used to modify the nature of the system[8].

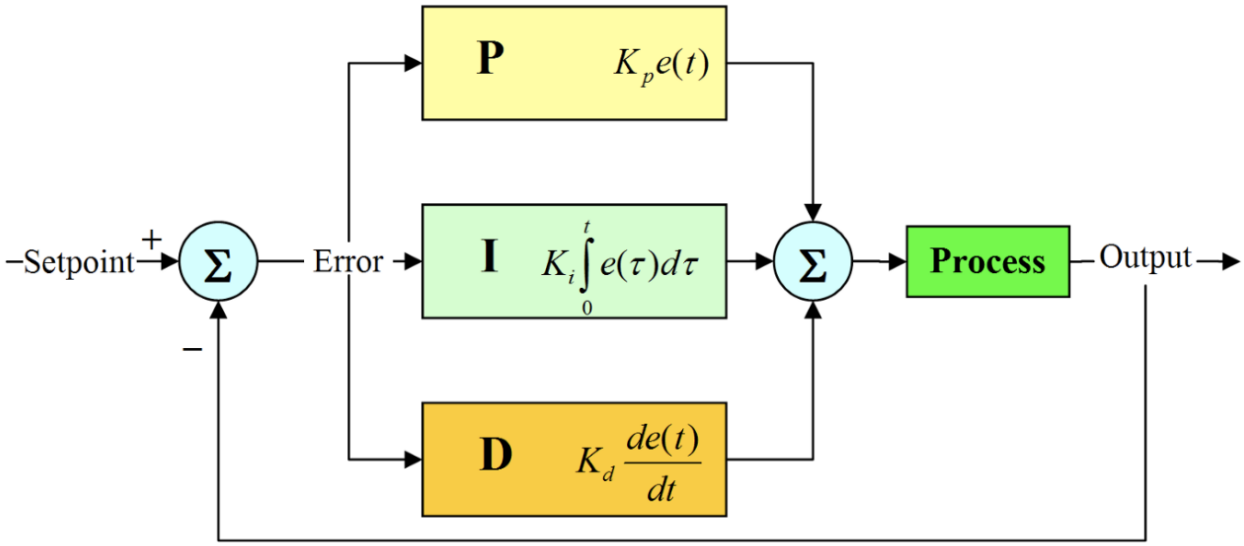

Fig. 3: PID controller with block diagram[3].

The calculation method of PID depends on three variables: Proportional, Integral, and Derivative. All three values are determined by the effect of the current error, the integral is the sum of the elapsed errors, and the derivative is the rate of change of the error value. We use the weights of these three to optimize the process by adjusting the built-in PID so that the control pattern can be adjusted to suit the process needs. A control's response takes the form of a controller motion that reaches an error value called overshoots and the oscillation of the system called oscillation If we were to write a simple equation of PID, we would get MV(t) = P_{out} + I_{out} + D_{out}. where value P_{out} comes from the error proportion term. The proportional response can be achieved by multiplying the constant error value.

$$P_{\text{out}} = K_{\text{p}} e(t)$$

Fig. 4: Calculation of Proportional term[4].

where P_{out} is the output signal, K_p is the proportional or variable gain, e is the error, and t is the time. If the output value is high, it means the error has changed a lot, while if the output is low, the error will change slightly. After that, we find the value by I_{out} is the result of the integral term being proportional to the magnitude of the error and the distance of the error over all time intervals, where the error increases progressively by the integral gain.

$$I_{\text{out}} = K_{\text{i}} \int_0^t e(\tau)\, d\tau.$$

Fig. 5: Calculation of Proportional term[4].

I_{out} The output signal of the integral term. ki is the integral gain, e is the error, t is the time, T is the integral variable. It will take the remaining error value of P_{out}. let's deal again but because it took the error value of Pout Let's make it possible for overshoots to happen.

$$D_{\text{out}} = K_{\text{d}} \frac{de(t)}{dt}$$

Fig. 6: Calculation of Proportional term[4].

D_{out} is the rate of error change resulting from the error each moment multiplied by the derivative gain. where the equation is as follows: k_d is the output of the differential term, D_{out} is the derivative gain, e is the error, t is the time.

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

Fig. 7: Calculation of PID-controlled[4].

and when P_{out}, I_{out}, D_{out} are combined, it will be a PID-controlled output signal. where the symbol u(t) is the output symbol and there are equations including P_{out},I_{out} ,D_{out} as follows:

3. Fuzzy logic

Fuzzy logic is a concept of logical analysis. which is different from the old logic that we are familiar with Usually there is only right and wrong, yes or no?, 0 or 1? by common sense If we can logically analyze events that are only right and wrong. It is considered that the logic is clear[9]. There is no ambiguity But there have been a number of incidents that, in fact, have troubled the analysts. What should be the logic? such as a medium gray color Say it's black, right? How will the reader respond? Or in that case, say that a 28-year-old lady is an adult? In these example cases, Analysts of different sexes, ages, and experiences will probably give different answers. Because there was a conflict in the answer itself. Since the events in this example do not clearly indicate one side, such as gray, it is neither white nor black. Whether the answer is white or black, it creates a conflict in the mind. Because there are only 2 possible answers. because traditional logic creates a framework thought like that So, analyzing the logic with these example cases thus leads to a new open logic presentation. wider than before and allow conflicts in the answers mentioned above to occur. to reflect reality as much as possible About 40 years back, L.A. Zadeh introduced the concept of fuzzy logic for describing events for analyzing the logic of events that may have some degree of conflict. Controversial or ambiguous In terms of giving logic, such as medium-light gray Say it's black, right? A fuzzy answer may have an answer that is 50% black, and 50% white, which each person may answer differently. You can see that the answer is clearly torn from the original idea is like this Because people's feelings are sometimes unpredictable. lack of certainty, Therefore, fuzzy logic is a solution that feels more reflective than traditional logic. mathematically Logical indication of events such as black and white, cool-fit-warm will be created as a function or set for describing the probability of an event What is the value in the range of 0-100%? This created set is called a Fuzzy Set. by creating multiple sets such as a set of white, black, or a set of cold air, a set of the right air, a set of warm air, etc. an event can be a member of every set, e.g. when the weather "starts to" sweltering, probability it's cold = 0%, probability it's fit = 20%, probability it's warm = 100%, etc. depending on with how to define these functions or sets This is the conceptual starting point for a new form of logic known as fuzzy logic. In a manner similar to the conventional logic with the words "yes" "or" "no" when it can be diagnosed. therefore proceed with one of the following actions. The engineering Commonly used fuzzy logic to help Analyze events to make decisions Applying to problems that are fuzzy or have high uncertainty The basic structure of fuzzy processing consists of four important parts as follows (Fig. 8)

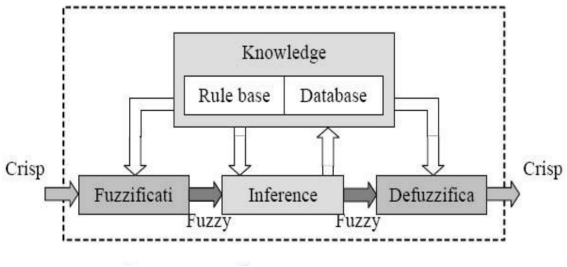

Fig. 8: fuzzy computing infrastructure[5].

Section 1 converts a common input to a fuzzification input, or in a fuzzy set, also known as a linguistic. Variable in the next section, Knowledge base, is the part where data is stored in two-component controls are Rule base and Database. In the Rule base subsection, the control method is defined. which is obtained from experts in the form of Linguistic rule data set and in the subsection Database This prepares the necessary parts in order to define the control rules and data manipulation of fuzzy logic The next section, the Inference Engine, is a fact-checking section and rules for interpreting reasoning as a mechanism for controlling the use of knowledge in problem-solving. including defining methods of interpretation to find answers The last section is Defuzzification which converts the output to the proper range. by making a conversion Information in fuzzy format is given as summaries or system control values. There are 4 parts of the fuzzy working process as follows (Fig. 9).

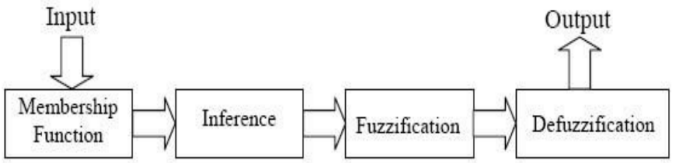

Fig. 9: Fuzzy logic processing steps[5].

The first step is to convert a variable input to a fuzzy variable input and create membership functions without needing must be of the same nature Depending on the characteristics of each input and the importance of the output interestingly, the function will look like a common language define. To make it fuzzy input, as shown in the picture (Fig. 10)

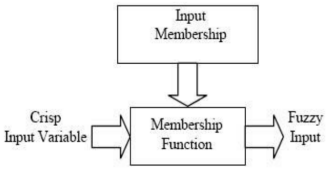

Fig. 10: The first step is fuzzy logic processing[5].

The second step is to establish the relationship between all the inputs related to the output at Relying on the principle of cause and effect may generate predictive data collection from the

decision of the human or experimental values by writing a system control rule Which will look in the form If ,and, or, which is a common language, all rules are processed together. in order to find a suitable decision, as shown in the picture (Fig. 11)

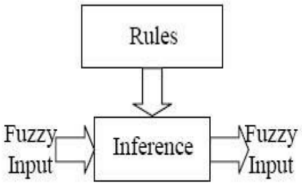

Fig. 11: The second step is fuzzy logic processing[5].

Step 3 is to find the fuzzy output by applying the control rules created in step 2 and processing it to the fuzzy input. using mathematical methods The value that has been processed as shown in the picture (Fig. 12)

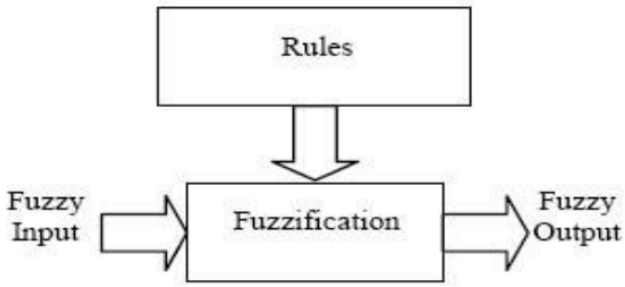

Fig. 12: The third step is fuzzy logic processing.[5].

How to make ambiguous values (Fuzzification) The popular methods for interpreting reasoning to use the Max-Min method and the Max-Dot method. Step 4 is the final step or the fuzzy reasoning step by changing the fuzzy output. to the output discipline and by mathematical methods such as How to find the Central of Gravity in order to use the values obtained in decision-making to control the system in that situation.

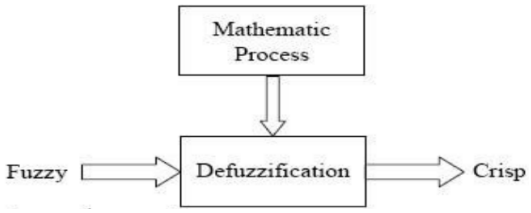

Fig. 13: The fourth step is fuzzy logic processing[5].

How to normalize fuzzy values (Defuzzification) method is a technique for selecting the maximum value. or summarizing the reasons from multiple sets to only one value This uses the maximum value of the membership level value. from multiple actions and choose only one action The Center of Gravity (COG) method is a method for averaging the results of interpreting The reason why it is used nowadays is, The resulting value to calculate the total center of gravity is obtained by estimating it from the equation.

$$COG = \frac{\sum_{i=1}^{N} \alpha_i w_i}{\sum_{i=1}^{N} \alpha_i}$$

Fig. 14: Equation for finding the Center of Gravity (COG)[5].

where each value of the equation is COG is the central of gravity, a_i is the fuzzy of the output at i, and finally w_i is the area under the fuzzy curve at i.

4. Graph

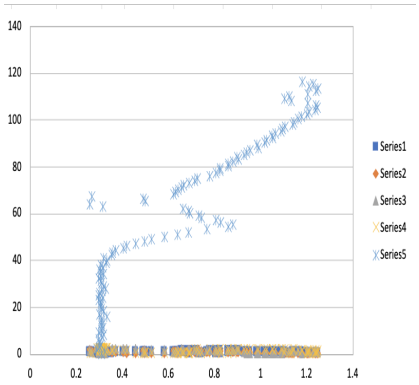

Fig. 15: laserData

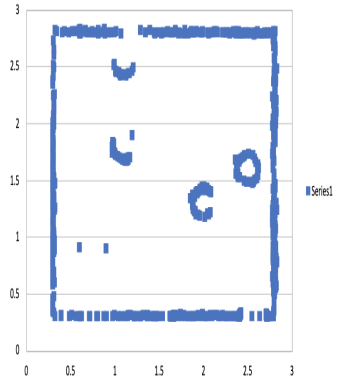

Fig. 16: laserMapData

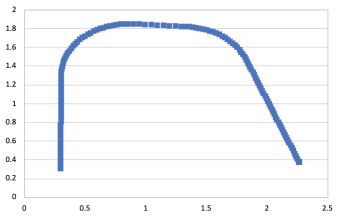

Fig. 17: odomTrajData

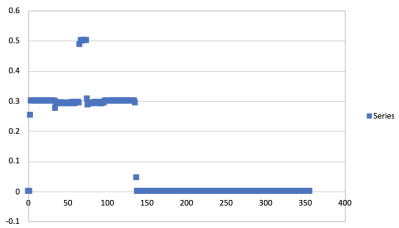

Fig. 18: odomVelData

Check robot movement in laser and odom

5. Appendix

5.1. PID controller for robot navigation

```
#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 "geometry_msgs/msg/pose.hpp"
#include "nav_msgs/msg/odometry.hpp"
#include <fstream>
#include <time.h>
#include <iomanip>

#include "sensor_msgs/msg/laser_scan.hpp"
using namespace std::chrono_literals;
using namespace std;
ofstream odomTrajFile; // Declare a file object to record odometry data.
ofstream odomVelFile;
ofstream laserFile; // Declare a file object for recording your laser data.
ofstream laserMapFile;s
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 – measureData;
        double ei = pid.ei_pre + err;
        double ed = err - pid.ed_pre;
        double output = kp*err + ki*ei + kd*ed;
        if (output > pid.Max_outout )
         output = pid.Max_outout;
        else if(output < -pid.Max_outout)
         output = -pid.Max_outout;
        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;
}

class Stopper : public rclcpp::Node{
public:
	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;

	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(&Stopp
er::odomCallback, this, std::placeholders::_1));

laserScan_=this->create_subscription<sensor_msgs::msg::LaserScan>("scan",10,std::bind(&St
opper::scanCallback, this, std::placeholders::_1));
	};
```

```
        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 robVelocity;
        int numberOfCycle=0;

        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);
        double laser_landmark1 = 1.3, laser_landmark2 = 1.4;
        double laser_landmark3 = 0.6, laser_landmark4 = 0.28, laser_landmark5 = 0.3;

        int stage = 1;
        void PID_wallFollowing(double forwardSpeed, double laserData);
        void PID_pass1stGap(double moveSpeed, double robotHeading);
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.
	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);
}

double odom_landmark3=1.20, odom_landmark4=1.80, odom_landmark5=2.25;


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(),"RobotPosfrontRange > laser_landmark3)tion: %.2f ,
%.2f",PositionX, PositionY );
	RCLCPP_INFO(this->get_logger(), "Robot stage: %d ", stage );
	odomTrajFile<< PositionX <<" "<< PositionY<<endl;

	robVelocity = odomMsg->twist.twist.linear.x;
	odomVelFile << numberOfCycle++ << " " << robVelocity << 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;
	controller.ed_pre=0, controller.ei_pre=0;
	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 = 0.1, 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::PID_pass2ndGap(double moveSpeed, double robotHeading)
{
	PID_para controller;
	double robotHeadingGap1 = 0; // the robot heading should be 0 degree
	controller.kp = 0.3, controller.ki = 0.04;
	controller.kd = 0.002, 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::PID_pass3rdGap(double moveSpeed, double robotHeading)
{
	PID_para controller;
	double robotHeadingGap1 = 0; // the robot heading should be 0 degree
	controller.kp = 0.5, controller.ki = 0.02;
	controller.kd = 0.003, 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::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 (PositionY < odom_landmark1 && PositionX < odom_landmark1a)
				PID_wallFollowing(FORWARD_SPEED_MIDDLE, leftRange);
			else stage = 2;
			break;


		case 2:
			if (PositionX < odom_landmark2)
				PID_pass1stGap(FORWARD_SPEED_MIDDLE,
robotHeadAngle);
			else stage = 5;
			break;
		case 3:
			if (PositionX < odom_landmark3)
				PID_pass2ndGap(FORWARD_SPEED_MIDDLE,
robotHeadAngle);
			else stage = 5;
			break;
		case 4:
```

```
                        if (PositionX < odom_landmark4)
                                PID_pass3rdGap(FORWARD_SPEED_MIDDLE,
robotHeadAngle);
                        else stage = 5;
                        break;

                case 5:
                        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;
}

//add the following code
void Stopper::startMoving(){

odomTrajFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/odomT
rajData.csv",ios::trunc); //note you should modify hhu to your username

odomVelFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/odomV
elData.csv", ios::trunc);

laserFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/laserData.c
sv",ios::trunc);

laserMapFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/laserM
apData.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();
        odomVelFile.close();
        laserFile.close();
        laserMapFile.close();
}

int main(int argc, char *argv[]){
        rclcpp::init(argc, argv);
        Stopper stopper;
        stopper.startMoving();
        return 0;
}
```

5.2.Fuzzy Controller for Robot Navigation

```
#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 "geometry_msgs/msg/pose.hpp"
#include "nav_msgs/msg/odometry.hpp"
#include <fstream>
#include <time.h>
#include <iomanip>

#include "sensor_msgs/msg/laser_scan.hpp"
using namespace std::chrono_literals;
using namespace std;

ofstream odomTrajFile; // Declare a file object to record odometry data.
ofstream odomVelFile;
ofstream laserFile; // Declare a file object for recording your laser data.
ofstream laserMapFile;

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 – measureData;
	double ei = pid.ei_pre + err;
	double ed = err - pid.ed_pre;
	double output = kp*err + ki*ei + kd*ed;
	if (output > pid.Max_outout )
	 output = pid.Max_outout;
	else if(output < -pid.Max_outout)
	 output = -pid.Max_outout;
	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;
}

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

        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(&Stopp
er::odomCallback, this, std::placeholders::_1));

laserScan_=this->create_subscription<sensor_msgs::msg::LaserScan>("scan",10,std::bind(&Sto
pper::scanCallback, this, std::placeholders::_1));
        };

        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 robVelocity;
        int numberOfCycle=0;

        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);
        double laser_landmark1 = 1.3, laser_landmark2 = 1.4;
        double laser_landmark3 = 0.6, laser_landmark4 = 0.28, laser_landmark5 = 0.3;

        int stage = 1;
        void PID_wallFollowing(double forwardSpeed, double laserData);
        void PID_pass1stGap(double moveSpeed, double robotHeading);
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::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::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::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.
        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);
}

double odom_landmark3=1.20, odom_landmark4=1.80, odom_landmark5=2.25;
```

```
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(),"RobotPosfrontRange > laser_landmark3)tion: %.2f ,
%.2f",PositionX, PositionY );
	RCLCPP_INFO(this->get_logger(), "Robot stage: %d ", stage );
	odomTrajFile<< PositionX <<" "<< PositionY<<endl;

	robVelocity = odomMsg->twist.twist.linear.x;
	odomVelFile << numberOfCycle++ << " " << robVelocity << 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;
	controller.ed_pre=0, controller.ei_pre=0;
	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 = 0.1, 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::PID_pass2ndGap(double moveSpeed, double robotHeading)
{
        PID_para controller;
        double robotHeadingGap1 = 0; // the robot heading should be 0 degree
        controller.kp = 0.3, controller.ki = 0.04;
        controller.kd = 0.002, 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::PID_pass3rdGap(double moveSpeed, double robotHeading)
{
        PID_para controller;
        double robotHeadingGap1 = 0; // the robot heading should be 0 degree
        controller.kp = 0.5, controller.ki = 0.02;
        controller.kd = 0.003, 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::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 (PositionY < odom_landmark1 && PositionX < odom_landmark1a)
                                Fuzzy_wallFollowing(leftRange, mleftRange);
                        else stage = 2;
                        break;


                case 2:
                        if (PositionX < odom_landmark2)
                                Fuzzy_to1stGap(leftRange, mleftRange);
                        else stage = 6;
                        break;
                case 3:
                        if (PositionX < odom_landmark3)
                                PID_pass2ndGap(FORWARD_SPEED_MIDDLE,
robotHeadAngle);
                        else stage = 5;
                        break;
                case 4:
                        if (PositionX < odom_landmark4)
                                PID_pass3rdGap(FORWARD_SPEED_MIDDLE,
robotHeadAngle);
                        else stage = 5;
                        break;

                case 5:
                        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;
}

//add the following code
void Stopper::startMoving(){

odomTrajFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/odom
TrajData.csv",ios::trunc); //note you should modify hhu to your username

odomVelFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/odomV
elData.csv", ios::trunc);

laserFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/laserData.c
sv",ios::trunc);

laserMapFile.open("/ufs/servg02/users/sk21395/M-Drive/ros_workspace/src/tutorial_pkg/laserM
apData.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();
	odomVelFile.close();
	laserFile.close();
	laserMapFile.close();
}

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

```
        rclcpp::init(argc, argv);
        Stopper stopper;
        stopper.startMoving();
        return 0;
}
```

6. Reference List

[1]Simon, b., (2017). *Hermes begis trial using self-driving robots in london* <https://www.bbc.co.uk/news/technology-39589967>

[2]Lilly C.,(2021). *How Does Spot Work? The Robot That Compares Design to Reality at the Construction Site* <https://www.archdaily.com/954784/how-does-spot-r-work-the-robot-that-compares-design-to-reality-at-the-construction-site>

[3]Arturo U.,(2011). *A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV).* <https://en.wikipedia.org/wiki/PID_controller#/media/File:PID_en.svg>

[4]Jaren c.,(2013)*pid Controller* <http://www.9engineer.com/index.php?m=article&a=show&article_id=2311>

[5]Damin p.,(2019) *fuzzy logic* <https://angsila.cs.buu.ac.th/~phong/Fuzzy/fuzzylogic.pdf>

[6]Aye Aye N, Aye Aye Z, and Wai Phyo Au.,(2008) *Control System Consideration of IR Sensors based tricycle Drive Wheeled Mobile Robot <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.307.6484&rep=rep1&type=pdf>*

[7]W. Kim.,(2005) *On target tracking with binary proximity sensors* <https://ieeexplore.ieee.org/abstract/document/1440939/authors#authors>

[8]Kiam Heong A.,(2005) *PID control system analysis, design, and technology* <https://ieeexplore.ieee.org/abstract/document/1453566>

[9]Petr H.,(2005) *Fuzzy logic from the logical point of view* <https://link.springer.com/chapter/10.1007/3-540-60609-2_2?noAccess=true>