Exploration and Object Trailing Strategies and Their Evaluation Using e-puck2 Microbots. (ACS6501)

Pranav Pramod Patil, Yifan Zhu and Lingyue Pan.

***Abstract*— The purpose of this study is to train and test the e-puck2 robot's ability to navigate its environment while avoiding obstacles and pursuing an object in an open space without colliding with it. These tests were conducted to assess the functioning of the robot, which alters its state using the eight infrared sensors on its circumference whilst maintaining a set speed. It was noted that the robot was able to perform these tasks satisfactorily, but it was unstable while turning.**

I. STRATEGIES

A. Obstacle avoidance and space exploration by the robot.

The aim of this task is to establish and evaluate how the robot will move in an environment containing obstacles while avoiding collision with the environment's limits.
Eight infrared sensors which are placed around the robot's circumference are used to carry out this task. Figure 1 shows the strategy for this task.

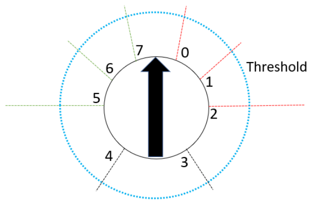

Fig. 1. Zone of bot motion.

The 8 infrared sensors are used to generate a threshold distance first. The front six infrared sensors are then used to scan the robot's field of view. The robot is programmed to advance forward using these sensors unless it senses an obstruction. If not, it comes to a stop and changes course under the control of the sensor it detected. To explain, the sensor that detects the greater distance from the threshold determined which way the robot would turn.

The robot changes its state while moving ahead when the front sensors detect an obstacle within the proximity range, and it is crucial that the front sensors do not detect any object within the proximity threshold range for the robot to move forward after turning.

The bot is designed to move at a specific speed during all of these tasks to ensure smooth operation.

Patil, Zhu and Pan are with the Department of Automatic Control and Systems Engineering, The University of Sheffield, UK {pppatil1, yzhu169, lpan15}@sheffield.ac.uk

B. Trailing an object.

The aim of this task is to follow a target object whenever it is within the sensing range or the robot's defined range. Furthermore, the robot must always keep a safe distance from the target at all times. To accomplish this task, a threshold value for the proximity sensors is set to prevent the robot from colliding with the obstacle.

To begin, the robot's 8 infrared sensors are programmed to scan for the object. If any of the sensors detect the object at any time, the robot's state changes. After sensing the object, the bot moves forward and chases it until it stops and changes its path, which changes the robot's state. When the object changes direction, the robot uses proximity sensors to follow it. When the object approaches the robot and crosses the threshold value, the robot is programmed to begin moving backward to avoid colliding with the object.

II. IMPLEMENTATION

A. Task 1: Obstacle avoidance and space exploration by the robot.

Different functions are constructed and initialized in order to ensure the correct order of actions for this task. (lines 3 – 42.)

The threshold limit for each and every one of the active infrared sensors is initially set to 300 (line 52). Additionally, the robot is programmed to move forward and turn at a speed of 800 steps per second. (lines 71, 72, 83, 84, 95, 96, 105, and 106.) Figure 2 shows a sequence of operations followed for this task.

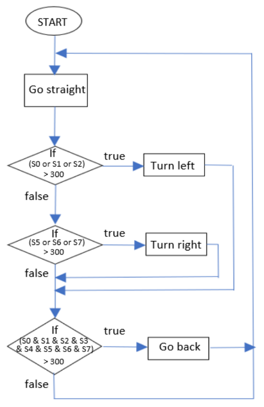
START
Go straight
If (S0 or S1 or S2) > 300
true
Turn left
false
If (S5 or S6 or S7) > 300
true
Turn right
false
If (S0 & S1 & S2 & S3 & S4 & S5 & S6 & S7) > 300
true
Go back
false

Fig. 2. Flowchart indicating sequence of operations of the bot for task 1.

The robot is programmed to perform the following tasks based on the conditions mentioned below.

a) Moving forward:
When all of the 8 infrared sensors do not detect the object, the robot is programmed to go straight switching on LED 1. (lines 93 – 101.)
b)Turning left:
When sensors 0, 1, or 2 detect an obstacle within the threshold range, a program is written to cause the robot to turn left, turning on LED 7. (lines 68 – 77.)
c) Turning right:
The robot turns right activating LED 3 when the value of sensor 5 is greater than 500 or either one of the sensors 7 or 6 identifies the obstacle in the threshold range. (lines 81 – 90.)
d) Moving back when approaching a dead end:
When the robot is nearing a dead end, and sensors 0, 1, 2, 5, 6, and 7 get activated at the same time, it turns around and goes to the start position, turning on LED 5. (lines 103 – 112.)

B. *Task 2: Trailing an object.*

The motors are first initialized, and the proximity sensors are calibrated. (lines 169 – 183.) The following step is to specify various threshold values. The target proximity threshold is set to 60, while the inner (target too close) and outer (target out of range) thresholds of the object position are set to 50 and 10, respectively.

The robot's speed is set at 600 steps per second when moving forward and backward, and 800 steps per second when changing its state to turn left or right. Figure 3 shows a sequence of operations followed for this task.

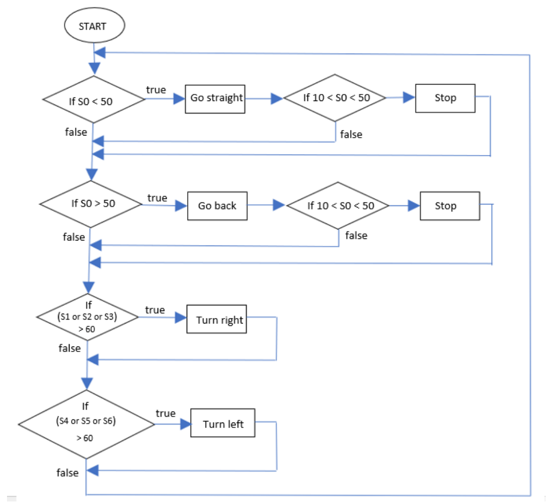

Fig. 3. Flowchart indicating sequence of operations of the bot for task 2.

A set of conditions are implemented for achieving the tasks mentioned below.
a) Moving straight: When sensor 0's value is less than 50, the robot is programmed to move forward by turning on LED 1. When the value of sensor 0 is between 10 and 50, the robot immediately stops. (lines 188 – 201.)
b) Moving back: A program is fed to the robot to move it in the reverse direction whilst turning LED 5 on when the sensor 0 value exceeds 50, whereas it is made to stop when the sensor 0 value lay in the range of 10 and 50. (lines 202 – 215.)
c) Turning right: When proximity sensors 1 or 2 or 3 exceed the threshold value of 60, the robot turns right, turning on LED 3. (lines 217 – 225.)
d) Turning left: When the reading of sensors 4 or 5 or 6 is greater than the threshold value of 60, the robot is instructed to turn left, activating LED 5. (lines 227 – 235.)

To compensate for noisy fluctuations in sensor values, cases a) and b) are given a range value of 10 – 50 rather than a specific value.

III. RESULTS AND DISCUSSIONS

A. *Task 1: Obstacle avoidance and space exploration by the robot.*

For this task, the robot was made to explore an enclosed environment avoiding obstacles at the same time.

Although the robot moved at a reasonable speed and did not collide with the wall boundaries despite being forced to pass through a narrow passage, it was unable to fully explore the environment. The bot became unstable while interacting with the object and the obstacles. Although it moved at a reasonable pace, its movement at turns was observed to be abrupt.

Based on the observations, the bot's turning instability could be improved by causing it to stop for a second and then reduce its speed before changing its state. To fully explore the environment, an algorithm should be developed in which the robot changes its direction at random after each cycle of operation.

B. *Task 2: Trailing an object.*

For this task, the bot was made to follow an object in an open environment without colliding with it.

It was noted that the robot was able to approach the object at a steady speed and also followed the object as it turned left and right. When the object approached the robot, it moved backward avoiding collision.

When the object was placed behind the bot, it was unable to detect it. Furthermore, when the object came to a complete stop, the bot moved back and forth, resulting in an abrupt movement.

According to the results, none of the sensors were activated when the object was placed behind the bot. This could be made possible by including a condition that causes the bot to rotate around itself, resulting in the bot detecting the object.

APPENDIX

```c
************   Task 1: Obstacle avoidance and space exploration by the robot.   **************

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "ch.h"
#include "hal.h"
#include "memory_protection.h"
#include <main.h>


#include "leds.h"
#include "spi_comm.h"
#include "sensors/proximity.h"
#include "motors.h"
#include "sensors/VL53L0X/VL53L0X.h"


messagebus_t bus;
MUTEX_DECL(bus_lock);
CONDVAR_DECL(bus_condvar);

//declare function
//int max(int num1, int num2, int num3, int num4, int num5, int num6, int num7, int num8);

int main(void)
{

    halInit();
    chSysInit();
    mpu_init();
   /** Inits the Inter Process Communication bus. */
  messagebus_init(&bus, &bus_lock, &bus_condvar);
  clear_leds();
  set_body_led(0);
  set_front_led(0);
    motors_init(); //motor initialization
    proximity_start(); //Start the proximity measurement module
    calibrate_ir(); //Calibrate the proximity sensors
    VL53L0X_start(); //


//    left_motor_set_speed(0); //left motor speed
//    right_motor_set_speed(0);  //left motor speed



  //  int get_prox(unsigned int sensor_number); // sensor_number 0-7

    int threshold = 300;   //set threshold for proximity(distance)
    int count = 0;
    /* Infinite loop. */
    while (1)
    {
      //waits 1 second
        //chThdSleepMilliseconds(1000);


       int proximity_reading[8] = { 0,1,2,3,4,5,6,7 };
              for (int i = 0; i < 8; i++)
              {
                  proximity_reading[i] = get_calibrated_prox(i);
              }
                //without distance sensor

              if (proximity_reading[0] > threshold || proximity_reading[1] > threshold || proximity_reading[2] > 500 )
              {
                  // chThdSleepMilliseconds(1000);
                  left_motor_set_speed(-800);
                  right_motor_set_speed(800); // if sensor 0 or 1 or 2 detects obstacle, robot turns left.
                  set_led(LED7, 1);
                  set_led(LED1, 0);
                  set_led(LED3, 0);
                  set_led(LED5, 0);
              }

                // turning while dis_sensor and pro_sensor approaching at the same time

              else if (proximity_reading[7] > threshold || proximity_reading[6] > threshold || proximity_reading[5] >
     500)
              {
                  left_motor_set_speed(800);
                  right_motor_set_speed(-800);//if sensor 7 or 6 or 5 detects obstacle, robot turns right.

                  set_led(LED3, 1);
                  set_led(LED1, 0);
                  set_led(LED5, 0);
                  set_led(LED7, 0);
              }

              else
              {

                  left_motor_set_speed(800);
                  right_motor_set_speed(800);//sensor 0, 1, 2, 3, 4, 7, 6, 5 don't detect the obstacle, robot goes
     straight.
            set_led(LED1, 1);
            set_led(LED3, 0);
            set_led(LED5, 0);
            set_led(LED7, 0);
                }

         if ( proximity_reading[0] > threshold && proximity_reading[1] > threshold && proximity_reading[2] > threshold
     && proximity_reading[7] > threshold && proximity_reading[6] > threshold && proximity_reading[5] > threshold)
              {
                  left_motor_set_speed(800);
                  right_motor_set_speed(-800);//If entering a dead end,  proximity sensor 0,1,2,5,6, and 7 detect an
     obstacle at the same time, the robot turns around and goes to starting point.
                  chThdSleepMilliseconds(1000);//wait 1s;
                  set_led(LED5, 1);
                  set_led(LED1, 0);
                  set_led(LED3, 0);
                  set_led(LED7, 0);
              }


             }



#define STACK_CHK_GUARD 0xe2dee396
uintptr_t __stack_chk_guard = STACK_CHK_GUARD;

void __stack_chk_fail(void)
{
    chSysHalt("Stack smashing detected");
}


************************         END OF TASK 1          ******************************



************************   Task 2: Trailing an object   ******************************



#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "ch.h"
#include "hal.h"
#include "memory_protection.h"
#include <main.h>


#include "leds.h"
#include "spi_comm.h"
#include "sensors/proximity.h"
#include "motors.h"
#include "sensors/VL53L0X/VL53L0X.h"


messagebus_t bus;
MUTEX_DECL(bus_lock);
CONDVAR_DECL(bus_condvar);

int main(void)
{

    halInit();
    chSysInit();
    mpu_init();
   /* Inits the Inter Process Communication bus. */
```

```
  messagebus_init(&bus, &bus_lock, &bus_condvar);
  clear_leds();
  set_body_led(0);
  set_front_led(0);
    motors_init(); //motors initialization
    proximity_start(); //Start the proximity measurement module
    calibrate_ir(); //Calibrate the proximity sensors
    VL53L0X_start(); //Initialize distance sensors

    /* Infinite loop. */

    //waits 1 second
    //chThdSleepMilliseconds(1000);

    int proximity_reading[8] = {0,1,2,3,4,5,6,7};
    for(int i = 0; i < 8; i++)
    {
        proximity_reading[i] = get_calibrated_prox(i);
    }


    int threshold_2 = 60;  //set threshold for proximity(distance)

    if(proximity_reading[0] < 50)
    {
        left_motor_set_speed(600);  // go straight
        right_motor_set_speed(600);
        set_led(LED1, 1);
        set_led(LED3, 0);
        set_led(LED5, 0);
        set_led(LED7, 0);
        if(proximity_reading[0] < 50  && proximity_reading[0] > 10)
        {
            left_motor_set_speed(0);  // stop if value > 50
            right_motor_set_speed(0);
        }
    }
    if(proximity_reading[0] > 50)
    {
        left_motor_set_speed(-600); // move back
        right_motor_set_speed(-600);
        set_led(LED1, 0);
        set_led(LED3, 0);
        set_led(LED5, 1);
        set_led(LED7, 0);

        if(proximity_reading[0] < 50  && proximity_reading[0] > 10)
        {
            left_motor_set_speed(0);  // stop
            right_motor_set_speed(0);
        }
    }
    if(proximity_reading[1] > threshold_2 || proximity_reading[2] > threshold_2 || proximity_reading[3] > threshold_2)
    {
        left_motor_set_speed(800);  // turn right
        right_motor_set_speed(-800);
        set_led(LED1, 0);
        set_led(LED3, 1);
        set_led(LED5, 0);
        set_led(LED7, 0);
    }

    if(proximity_reading[4] > threshold_2 || proximity_reading[5] > threshold_2 || proximity_reading[6] > threshold_2)
    {
        left_motor_set_speed(-800); // turn left
        right_motor_set_speed(800);
        set_led(LED1, 0);
        set_led(LED3, 0);
        set_led(LED5, 0);
        set_led(LED7, 1);
    }
}


#define STACK_CHK_GUARD 0xe2dee396
uintptr_t __stack_chk_guard = STACK_CHK_GUARD;

void __stack_chk_fail(void)
{
    chSysHalt("Stack smashing detected");
}
*************************   END OF TASK 2   ********************************
```