

CPE 470/670 Autonomous Mobile Robots

Professor: Dr. Monica Nicolescu

FINAL

Golf Contest

Team #1

Mubeen Parkar

Masahiro Kotani

Dec 18, 2005

1. Introduction

In this lab we prepared our robot for the golf Contest. The goal of our robot is to collect golf balls and deposit into a hole which is marked circular area in the middle of the arena. Robots play against each other in a double elimination format competition in games that last 2.5 minutes. If the team has two consecutive losses a robot is eliminated from the tournament.

We needed to design and build a new robot so that it will collect the ball. Also we need to keep the robot robust so that it won't break middle of the competition. We made many changes so that the robot will collect and not lose the ball without getting stuck in the wall. We constructed the robot from scratch atleast 3 times until we were satisfied with the physical design. We used portions of the programs we used in previous contests. Also we add additional features in our programs like gate open and gate close.

2. Hardware and software design

Our hardware is designed as follows. The optosensors were mounted in front left and front right of the robot so as to maximize the chances of detecting the hole. We have front bumper and side bumper to avoid obstacles. We fine tuned these bumpers to detect even a minor touch. When the front bumper is activated, the robot will avoid the obstacles. Side bumper was used because balls tend to sit on the wall later in the competition. Side bumper allow us to move circular is arena (wall following). We made the robot higher so that it will collect the balls smoothly. Gate is attached top of the robot and inside the bumper to avoid hitting obstacles. We tested the physical design to such an extent that we were absolutely sure that our robot would never get stuck at any edge of the competition arena.

Our software is designed as follows. We used the code from the previous labs. First, we use obstacle avoidance with gate open and close feature. Next thing we did was to detect the hole when the robot is wandering. We also added wandering program which will take random turn every 9 second. This timer helped the robot immensely as it never got stuck in any situation. Those were the basic robot movements. However we added some program, it will collect 6 balls at the default place near the robot and bring them to hole, so that we won't loose the match.

3. Problem encountered

- When the robot is facing opponent, we were not able to collect 6 balls first.

- The robot will not often go to the middle at the hole.
- The robot got stuck at corners.
- The robot couldn't get the ball out of the corners.
- Battery died faster than expected and the turning angle kept changing

4. Solutions to the problem

In order to solve the problem, we programmed the robot to wander around every 9 seconds. This was the solution of the problem of getting stuck, or not going to the middle hole. When the robot is facing opponent, we just go straight and try to grab their balls and come back to hole. This way we can mess up opponent's strategy. Battery died so fast so we had to give short charges in zap mode.

5. Unsolved problems

Since the wandering program is randomly generated, if something goes wrong, takes time to go to the middle hole. (happened during the competition).

6. Inference

We worked on the physical design and the code of the robot extensively. We eliminated many design and code problems. Hence overall our robot was robust. Our opponents relied heavily on their strengths and ignored their weaknesses in design and code and that caught them unaware in the competition. We would win the final match as final winners had 2 losses in the competition, both were due to us. In the final game due to very low battery power we lost to them by one golf ball.

7. Code

```
//CPE470/670 Group #1
//Mubeen Parkar
//Masahiro Kotani
//FINAL -Golf Contest-

int L_SENSOR = 6;    //line sensor on left
int R_SENSOR = 4;

int L_LINE_SETPPOINT = 190;    //Distinguish b/w black and white
int R_LINE_SETPPOINT = 190;

int LEFT_TOUCH = 11;    // touch sensor
int RIGHT_TOUCH = 10;

int RIGHT_MOTOR = 0;
```

```

int LEFT_MOTOR = 3;

int left_line_sensor()    //return line sensor reading
{
    return analog(L_SENSOR);
}

int right_line_sensor()   //return line sensor reading
{
    return analog(R_SENSOR);
}

int left_on_hole()        //opto sensor LEFT
{
    return(left_line_sensor()>L_LINE_SETPOINT);
}

int right_on_hole()       //opto sensor RIGHT
{
    return(right_line_sensor()>R_LINE_SETPOINT);
}

int left_right_on_hole()  //check if both sensor is black or not
{

return((left_line_sensor()>L_LINE_SETPOINT)|| (right_line_sensor()>R_LIN
E_SETPOINT));
}

float timepast;

void reset_timer() //reset timer for corner escape
{
    timepast = seconds();
}

float timegone()     //time counter for corner escape
{
    return seconds() - timepast;
    return seconds() - timepast;
}

void backward()
{
    motor(RIGHT_MOTOR, -60);
    motor(LEFT_MOTOR, -60);
}

void right()
{
    motor(RIGHT_MOTOR, 80);
    motor(LEFT_MOTOR, -50);
}

void left()
{
    motor(LEFT_MOTOR, 80);

```

```

        motor(RIGHT_MOTOR, -50);
    }

void forward()
{
    motor(RIGHT_MOTOR, 60);
    motor(LEFT_MOTOR, 60);
}

void stop()
{
    off(RIGHT_MOTOR);
    off(LEFT_MOTOR);
}

void left_avoid()
{
    backward();
    sleep(0.2);
    left();
    sleep(0.6);
}

void right_avoid()
{
    backward();
    sleep(0.2);
    right();
    sleep(0.6);
}

void turn90() //turn 90 degree to the left
{
    backward();
    sleep(0.2);
    motor(RIGHT_MOTOR, 100);
    motor(LEFT_MOTOR, -100);
    sleep(0.39);
    backward();
    sleep(0.1);
    stop();
    sleep(0.1);
}

void random_avoid() //corner escape with gate open/close
{
    gate_close();
    stop();
    sleep(0.2);
    backward();
    sleep(0.4);

    if (random(2) == 0)
    {
        left();
    }
}

```

```

        else
        {
            right();
        }
        sleep((float)random(100)/100. + 0.5);    //random time generator
b/w .5 and 1.49
        gate_open();
    }

void random_avoid_d() //corner escape without gate open/close
{
    backward();
    sleep(0.4);

    if (random(2)== 0)
    {
        left();
    }
    else
    {
        right();
    }
    sleep((float)random(100)/100. + 0.5);    //random time generator
b/w .5 and 1.49
}

void gate_close()
{
    motor(2,-100);
}

void gate_open()
{
    motor(2,100);
}

int z =1;    //time counter for corner escape
float time_step = 6.0;    //time range for corner escape

void main()
{
    int totalhits = 0;
    int prevhits = 0;
    int flag=1;

    reset_timer();
    gate_open();
    while(1)
    {
        while(!start_button())
        {
            stop();
        }
        reset_system_time();    // reset time to count corectly

        while(!stop_button())
        {

```

```

//Error Check
printf(" %f\n", seconds() );
sleep(0.1);

forward();

//if hits the wall within 1.5 second, reset time and turn
90 degree
if(seconds() < 1.5 &&
(digital(LEFT_TOUCH)||digital(RIGHT_TOUCH)))
{
    beep();
    turn90();
    reset_system_time();
}

//if hits the wall within 2sec and 3 sec go strait to hole
if((seconds() > 2.0 && seconds() < 3.0) &&
(digital(LEFT_TOUCH)||digital(RIGHT_TOUCH)))
{
    gate_close();
    backward();
    sleep(0.2);
    motor(RIGHT_MOTOR, 100);
    motor(LEFT_MOTOR, -100);
    sleep(0.43);
    gate_open();
}

//turn randomly every 9 second
if((int)seconds()%9==8)
{
    beep();
    gate_close();
    stop();
    sleep(0.2);
    random_avoid();
    sleep(0.5);
    gate_open();
}

//Error check
//printf("totalhits: %d\n", totalhits);
//printf("L:%d R:%d: \n", left_line_sensor(),
right_line_sensor());

if (digital(LEFT_TOUCH)) //obstacle avoidance
{
    gate_close();
    stop();
    sleep(0.2);
    left_avoid();
    if(prevhits ==0)
    {
        totalhits+=2;
    }
    totalhits++;
}

```

```

        prevhits =1;
        gate_open();
    }

    if (digital(RIGHT_TOUCH))          //obstacle avoidance
    {
        //beep();
        gate_close();
        stop();
        sleep(0.2);
        right_avoid();
        if(prevhits==1)
        {
            totalhits+=2;
        }
        totalhits++;
        prevhits =0;
        gate_open();
    }

    //time counter for corner escape
    if (timegone() >time_step)
    {
        reset_timer();
        z++;
        totalhits=0;
    }
    //hit counter for corner escape
    if (totalhits >3)
    {
        random_avoid();
        totalhits =0;
    }

    if(right_on_hole())          //detected hole on right
    {
        gate_close();
        left();
        while(left_right_on_hole());          //wait until both
sensor is on black
        stop();
        sleep(0.3);
        gate_open();
        backward();
        sleep(0.75);
        random_avoid_d();
    }

    if(left_on_hole())          //detected hole on left
    {
        gate_close();
        right();
        while(left_right_on_hole());          //wait until both
sensor is on black
        stop();
        sleep(0.3);
        gate_open();
    }

```



```
        backward();
        sleep(0.75);
        random_avoid_d();
    }
} //while loop for start button ends here
stop();
} //while loop for start or stop button ends here
} //main program ends here
```