Formal Report
Clover

Date: 4/19/11
Student Name **Ruibiao Song**

TAs: Josh Weaver
Tim Martin
Devin Hughes
Ryan Stevens
Sean Frucht
Instructor: Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

University of Florida

Department of Electrical and Computer Engineering

EEL 4665/5666

Intelligent Machine Design Laboratory

Table of contents

Abstract

Picking up the table tennis ball is a dull work after playing a game. My robot Clover will gather the table tennis ball lying on the ground to simplify this task. It will find the target and grab it to put it in his back basket. When all table tennis balls are finished, it will go back to a predefined spot. With the assistant of this robot, the play ground will be kept well organized and make the players enjoy their sports much better.

Executive Summary

The robot clover is a completely autonomous robot capable of navigating and making obstacle avoidance while searching for the target ball. The robot is a prototype built for the intelligent Machine Design Laboratory for academic credit through University of Florida in the spring of 2011 semester.

This autonomous vehicle uses vision from an IP camera to track a predefined color and then navigates to center the mass of pixels in the robot point of view. When the ball is close enough the robot will stop and the collecting ball program will be activated. Robot will lay down his arm and open the claw to grab the ball. When this process finished, it will lift his arm again to search another ball.

Two IR sensors were installed in left front and right front sides of the robot to give it the ability to avoid obstacles without coming in contact with them. The sonar was installed in middle front of the robot giving it ultrasonic sight in the direction it travels most of the time. When the camera senses a target ball of the correct color the sonar will be activated to measure the distance between the robot and the target ball.

Introduction

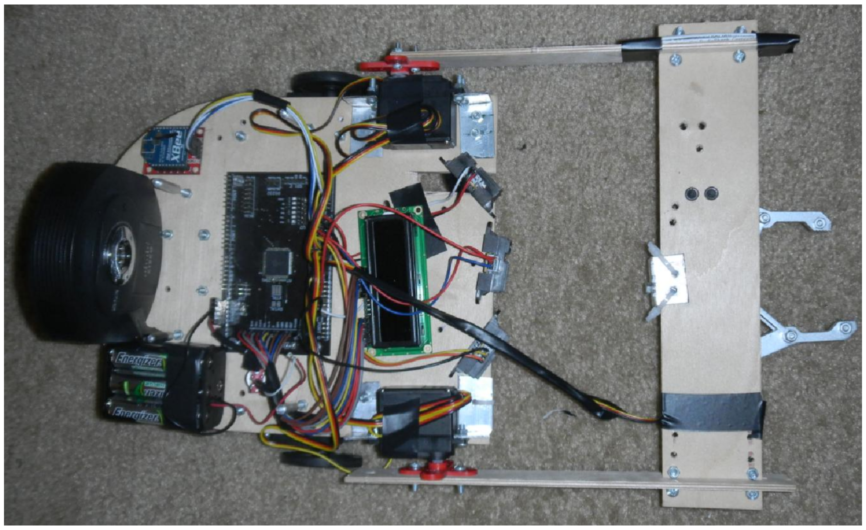
I always play table tennis during the weekend and use lot of balls, but picking up them by me is always a tough work. So I supposed to build a robot to assist me finish this task. The robot I designed will pick up balls in an efficient way and put them in his back basket, which enable the robot to store numbers of balls. The paper will begin by describing the project as a whole and then begin to detail each system.

Integrated System

In general, the robot will perform four functions in his autonomous manner. They are obstacles avoidance, table tennis balls seeking, picking up the table tennis ball, put the ball. Travelling forward, the robot will change its direction if the obstacle is not a target ball appears within a certain distance. Travelling along the tennis court, the robot will seek the tennis ball using the IP camera, and once the ball is found he will roll toward the target and picking up the ball use his claw to grab the ball. Once all the balls are collected, it will go to a specified spot autonomously.

The robot will use a Pridgen Vermeer Robotics Xmega128A1 microcontroller board to integrate and control all of its sensors and servos. The sensors and servos will be connected to the microcontroller board in order to ensure that the motors receive an adequate amount of voltage, and also that the current from the motors do not harm the board. The Xmega128A1 board also includes Data, ADC, PWM, and SCI ports that are useful for interfacing with sensors. The robot have two servos to control the movement of the robot, two servos used to lift the arm, one servo for open and close the claw to grab the ball. For sensors, two IR for obstacle avoidance, sonar for measure the distance and IP camera used for detect the ball.

Mobile Platform



The mobile platform took a long time to be built, especially the claw part, which is hard to assemble. I draw the platform of the robot using the

Solidworks and TA helps me cut the base platform using t-tech. Since there is always some uncertainty about the design, I drill the holes for assembling later to avoid the problem of unfitness.
My robot had four wheels two controlled by servos and two casters. Two servos could control the forward, backward, right turn and left turn of the robot, and two casters used for keep balance. The two servos connected to the arms are used for lift the claw. I have tried to use one servo to lift it up, but it's too heavy and now the two servos work well. For the servo connected to the claw, it was fixed on the front plate and the rotate of the servo will drive the gear of the claw open and close.

In retrospect to all the process I have experienced for building this model, it would have been better if I had taken the two or three weeks to built the complete structure of the robot in Solidworks, which will save me amounts of time
and properly model the robot. I thought was robot platform was too simple to r equire modeling on Solidworks and having a TA cut out the parts on the T-Tech machine. But I ended up spending most of my time getting the platform to wor k and it now looks like an hacked together jungle of wire.

Actuators

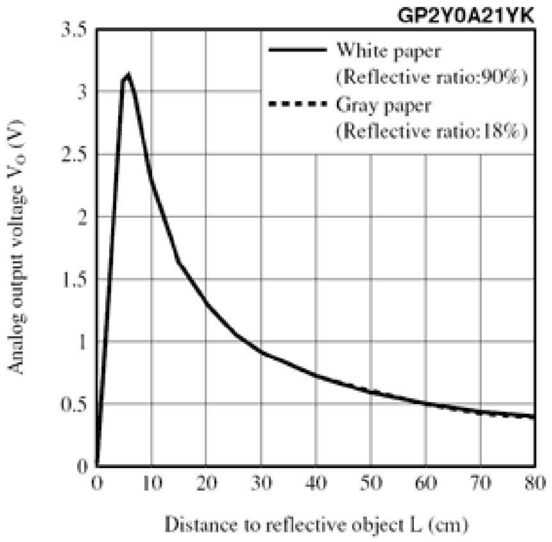
I used five HS-311 Standard servos. Two of them were hacked and to be used as wheels. Other there were to drive the arms and claw respectively.

Sensors

The sonar sensors, IR sensors and IP camera will outfit the sensor array on this robot. The bump sensor is mainly used for sensing whether the robot has grabbed the tennis ball and it will always send signal to the processor to inform that the picking up process is working well. Also the bump sensor is also used for the bump to stop working of the robot. The IP camera will be used for detecting the target balls. By tracking the fluorescent orange color of the ball, IP camera seems to be a better choice since the CMU cam is easy to be influenced by the other light.

Sharp GP2Y0A21YK0F Analog Distance Sensor 10-80cm

The Sharp distance sensors are a popular choice for many projects that require accurate distance measurements. This IR sensor is more economical than sonar rangefinders, yet it provides much better performance than other IR alternatives. Interfacing to most microcontrollers is straightforward: the single analog output can be connected to an analog-to-digital converter for taking distance measurements, or the output can be connected to a comparator for threshold detection. The detection range of this version is approximately 10 cm to 80 cm (4" to 32"); a plot of distance versus output voltage is shown below.



Bump

The bump switch mechanism on the robot will consist of a micro switch with an extended trigger mounted on the perimeter to aid in obstacle avoidance. The bump sensor will cause the robot to cut power to the motors and change direction in a random fashion. The goal to implement these sensors will be to mount them and to program a stop command to the algorithm controlling the motors.They are manufactured by Omron Electronics Part#SS-5GL2. The sensor is not applied to the robot as of this writing. Six total sensors will be mounted to allow the robot to navigate in its environment by touch. Normally a micro-switch is wired in parallel to the power supply and some sort of tool or device that requires a safety switch. When the switch is activated, the power to the device is cut off to stop the device the switch is wired to. When a switch is activated, it enables a pin on Port A on the microprocessor to be written high. The Pin is written high and this causes the motors to stop and then move in a random direction. No data base is needed on this sensor. It does not return values, so it is simply a switch to be turned on or off.

Sonar

The sonar sensors consist of sonic devices that will allow the robot to navigate beyond the touch of the bump sensors by measuring the robots distance away from an object in its path. A sonar sensor bounces sound pulses from an emitter and waits a short amount of time to receive the same pulse. The sonar sensor has three objectives to be met upon completion. The first is to wire them to the microcontroller and mount them. The second is to activate the sonar's ability to pulse through software. The third objective will be to integrate these sensors in to the main program for obstacle avoidance through external and internal interrupts. These sensors can be bought from:
Acroname Robotics.
Two Sensors are mounted on the front of the robot on either side at a 45 degree inward angle and set to pulse at two different times. This array arrangement should provide the robot with the maximum amount of "vision" with a minimum amount of sensors. This sensor works by sending out a high pitched pulse of sound activated from the trigger input pin, which is activated by a falling edge signal from the microcontroller. The pulse bounces off an object in its path and travels back to the sensor. After waiting 10 micro seconds, the sonar receives the same pulse and sends a signal back to the microcontroller through the Echo output pin. The microcontroller then calculates the amount of time has passed between the sonar's sending and receiving signals and outputs a distance to the LCD or a command to the motors. The sensor software algorithm uses ports D and A on the microcontroller, one of the microcontroller's 16 bit timers, TCNT1, an interrupt that counts the number of timer overflows, and an External interrupt activated upon the sonar's second pulse falling edge signal and uses the microcontrollers LCD on Port B to blink when ever an interrupt is fired. This algorithm still does not produce numbers that are consistently between the reasonable range of 0 to 255. Further debugging is needed before usable values can be obtained. Data values obtained as of this writing vary from 16 to 1600 nonlinearly, and seem to increase as an object gets closer. The sonar's current values seem to contradict to normally operating sonar's values.

Special Sensors

Cisco-Linksys Wireless-N Internet Home Monitoring Camera:

I bought this camera on Amazon. Cisco-Linksys Wireless-N Internet Home Monitoring Camera is a network camera which has web server function and enables to connect to the Internet itself. Since it supports dynamic body detection and has an alarm notification, it will be suitable for anticrime, security monitoring. Use web browser for monitoring your house from outside the home or set up at the front door to monitor. Since it has an independent system with internal CPU, it won't require the computer to send out the motion data. Only connecting to the network, it can be sent out the data and monitor. It has dynamic body detection which senses the motion within the rage of the camera. It enables to notify the change of image of the camera or send out the snap shpt by email or upload the images to FTP server. Besides this camera, I also bought XBEE online to transfer the information.
The website that I bought the camera is http://www.linksysbycisco.com/LATAM/en/products/WVC80N.

Behaviors

The robot will have 3 behaviors: avoid obstacles, seeks and picks up the tennis balls. All these behaviors should perform as soon as possible since players always use lot of balls.
The first behavior to be discussed is obstacle avoidance. When the robot detects the obstacle with in a certain distance, which is not the target, table tennis ball, he will change his direction. The behavior is based on the values the sensor return.
The second behavior will be seeking the ball. When the IP camera returns a signal that the target is a table tennis ball, he will travel directly forward using the coordinate in himself. And the signal of sonar decides how far the robot will move.
After travelling to front of the tennis ball, it will first close his front arm to grab the ball and then put his ball to his back basket.
When all the balls are collected, it will autonomously go back to a predefined

spot, which is usually the corner of the court to make it convenient for the players to fetch the ball.

Conclusion

Being mechanical engineering major, I had little knowledge of the program and electronics aspects. I tried to do a better job of the platform and structure design but to be as simple as possible for the programming but I have tried go with the herd as far as the robot parts were concerned. In retrospect tho ugh, I underestimated just the basic program rules of C and C++. For the fail of this project, I think its main reason it's the ability of the camera to detect the table tennis ball. Although he can find the tennis ball by using the color detect, the camera maybe affected by the environment, such as the sun light or other similar color. If I have tried to use the camera to detect the shape of the ball, it would be a better result. Due to my limit knowledge of programming, detecting the color takes me almost one month, so it would be really hard for me to find the solution for shape detect.

For all the days that I work with the robot, initial days were spent in apprehension of doing something wrong and frying u p the expensive electronic components. Many things were reworked and reworked until I got it right. I got little time to spend on the software aspect of the robot. In retrospect, it might have been actually better to get a prebuilt robotics kit and then tried to focus on implementing some advanc ed programming behaviors. That will definitely be the approach I take with my second robot.

Regardless I'm really glad I took this course and also I treasure the time of my hard work with the robot.
I know now how to go about building a robot and if I could take the course ove r again, I would make a great robot! I would avoid the mistakes that seem stupid in retrospect.

Thanks to:
Dr. Arroyo and Dr Schwartz – thanks for the opportunity
ALL the TAs thanks for helping me out of the troubled situations.
Mingwei Liu for solving the problem of code.

Documentation

References:

IP camera Cisco-Linksys Wireless-N Internet Home Monitoring Camera user guide.
http://downloads.linksysbycisco.com/downloads/userguide/1224644814946/WVC80N_V10_UG_NC-WEB.pdf

HITEC HS-311 datasheet.
http://www.lynxmotion.com/ghm02.htm

Devantech SRF05 UltraSonic Range Finder. © 1994-2007,
Acroname Inc.,

Sharp GP2Y0A02YK0F Distance Measuring Sensor Unit Measuring distance datasheet.

Appendices

Program code

Clover.c

```
#include <avr/io.h>
#include "PVR.h"
#include <avr/interrupt.h>
ISR(USARTF0_RXC_vect)
{
	RXdata= USARTF0_DATA;
	//Usart_PutChar(RXdata);
}

void main(void)
{
	//initialization
    xmegaInit();                    //setup XMega
    delayInit();                    //setup delay functions
    ServoCInit();                   //setup PORTC Servos
    SonarInit();                    //setup PORTD Servos
    ADCAInit();                     //setup PORTA analog readings
    lcdInit();                      //setup LCD on PORTK
    UsartInit();
    //ADCA0 RIGHT IR
    //ADCA1 LEFT IR

    //PWMD4 RIGHT ARM SERVO
    //PWMD5 LEFT ARM SERVO

    //PWMD3 CLAW SERVO
    //PWMC0 right wheel SERVO
    //PWMD1 left wheel SERVO

    //Claw close position servoc3(40)
    //Claw open position servoc3(-50)

    //right arm position servocd4(-60(moving)~100(working))
    //left arm position servocd5(-60(working)~100(moving))

int lf,rf,lb,rb;      //used for control velocity of the wheel
        int rv,lv;              //velocity of the wheel
        int k;          //used for control the claw angle
        int range;    //distance of the ir
        int right_ir;
        int left_ir;
        int n;
        int m;
        n=100;
        m=-60;
        lf=rf=20;
        lb=rb=-10;

        //set initial position
        while(n>-60)
        {
                ServoC5(m);
                ServoC4(n);
                delay_ms(50);
                n=n-2;
                                m=m+2;
        }
        ServoC3(-50);
          ServoC1(lf);
              ServoC0(rf);
              delay_ms(1000);

while(1)
                {
                left_ir=ADCA1()-200;
                left_ir=left_ir*0.5135/9.8;
                right_ir=ADCA0()-200;
                right_ir=right_ir*0.5135/9.8;
                range=ADCA3()-200;
```

```
range=range*0.5135/9.8;

/*do
        {
        range=(TCD0_CCA);
        }
while (range <= 700);
*/
if (range>=115 && range<=125)
{break;}

/*lcdGoto (0,0);
lcdInt (left_ir);
lcdGoto (1,0);
lcdInt (right_ir);
lcdGoto (1,5);
lcdInt (range);
delay_ms(500);
*/
if (left_ir > 170)
        rv=rb;
else
        rv=rf;

if (right_ir > 170)
        lv=lb;
else
        lv=lf;


ServoC1(lv);
ServoC0(rv);

delay_ms(1000);

if (RXdata == 'r')
        {

ServoC1(10);
ServoC0(20);
lcdGoto (0,0);
lcdString ("turn left");
                delay_ms(400);
        };
        if (RXdata == 'l')
            {
        ServoC1(20);
        ServoC0(10);
                lcdGoto (0,0);
        lcdString ("turn right");
                delay_ms(200);
            }

        lcdData(0x01);
        }

        ServoC1(0);
        ServoC0(0);

        while(n<100)
                    {
                    ServoC5(m);
                    ServoC4(n);
                    delay_ms(50);
                    n=n+2;
                    m=m-2;
                    }

       while(k<60)
                    {
                    ServoC3(k+10);
                    k=k+10;
                    delay_ms(100);
                    }
                    delay_ms(1000);
       while(n>-60)
                    {
                    ServoC5(m);
                    ServoC4(n);
                    delay_ms(50);
                    n=n-2;
                    m=m+2;
                    }
}
```

Image Processing Code:

```
#include "stdafx.h"
#include <stdio.h>
#include "opencv\cv.h"
#include "opencv\highgui.h"
#include "CnComm.h"

using namespace cv;
using namespace std;

// HSV - HUE (color), SATURATION (0 grey, 255 color), VALUE (0 black, 255 white)

int sut_r=255;
int slt_r=100;
int vut_r= 255;
int    vlt_r=100;

int g_switch_value = 0;

int filterIntr = 0;
int filterIntr2 = 0;
int lastfilterInt = -1;

void switch_callback_r( int position )
{
        filterIntr = position;
}
void switch_callback_r2( int position )
{
        filterIntr2 = position;
}

        //// Thresholding Function



IplImage* GetThresholdedImage_r(IplImage* img)
{
        // Convert the image into an HSV image
        IplImage* imgHSV = cvCreateImage(cvGetSize(img), 8, 3);
        cvCvtColor(img, imgHSV, CV_BGR2HSV);

        // Create new image to hold thresholded image
        IplImage* imgThreshed = cvCreateImage(cvGetSize(img), 8, 1);
        //IplImage* imgThreshed1 = cvCreateImage(cvGetSize(img), 8, 1);
        //cvInRangeS(img, cvScalar(0, 0, 120), cvScalar(100, 100, 255), imgThreshed); // apply threshold
        cvInRangeS(imgHSV, cvScalar(filterIntr2, slt_r, vlt_r), cvScalar(filterIntr, sut_r, vut_r), imgThreshed);
// apply threshold
        cvAdd(imgThreshed,imgThreshed,imgThreshed);

        cvReleaseImage(&imgHSV);
        return imgThreshed;
        //cvReleaseImage(&imgThreshed1);
        //cvReleaseImage(&imgThreshed);
}

class HelloComm : public CnComm
{       //! \sa CnComm::OnReceive()
        void OnReceive()
        {
        }
};

int main( int argc, char **argv )
{
        const char* name = "Filters Window";
        CvCapture *capture = 0;
        IplImage    *frame = 0;
        int             key = 0;

        /* initialize camera */
        //capture = cvCaptureFromCAM(-1);
        capture = cvCreateFileCapture("http://192.168.1.4/img/video.mjpeg");

        //capture = cvCreateFileCapture("http://10.0.0.5/img/video.mjpeg");
        //capture = cvCaptureFromCAM("http://192.168.1.67/img/video.mjpeg");

        HelloComm Com;
        Com.Open(4,9600);

        /* always check */
        if ( !capture )
        {
                fprintf( stderr, "Cannot!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! open initialize webcam!\n" );
                return 1;
        }

        /* create a window for the video */
```

```
cvNamedWindow( "Video", CV_WINDOW_AUTOSIZE );
cvNamedWindow( "Threshed", CV_WINDOW_AUTOSIZE );

 // This image holds the "scribble" data...
// the tracked positions of the ball
IplImage* imgScribble = NULL;

 cvNamedWindow( name, 1 );    // for trackbar??

 // Create trackbar

 cvCreateTrackbar( "MaxValHue_r", name, &g_switch_value, 255, switch_callback_r );
 cvCreateTrackbar( "MinValHue_r", name, &g_switch_value, 255, switch_callback_r2 );


while( key != 'q' )
 {
        /* get a frame */
        frame = cvQueryFrame( capture );

        /* always check */
        if( !frame ) break;

         //cvErode(frame, frame, 0, 2); // ADD this line
         //cvSmooth( frame, frame, CV_GAUSSIAN, 9, 9 );             /////    SMOOTHING!!

         // If this is the first frame, we need to initialize it
         imgScribble = NULL;
        if(imgScribble == NULL)
        {
               imgScribble = cvCreateImage(cvGetSize(frame), 8, 3);
        }

         // Holds the thresholded image (green/blue/red = white, rest = black)

        IplImage* imgRedThresh = GetThresholdedImage_r(frame);

         //// Calc position assuming its the only thing that is that color!

         // Calculate the moments to estimate the position of the ball

        CvMoments *moments_r = (CvMoments*)malloc(sizeof(CvMoments));
        cvMoments(imgRedThresh, moments_r, 1);
```

```
                double moment10_r = cvGetSpatialMoment(moments_r, 1, 0);
                double moment01_r = cvGetSpatialMoment(moments_r, 0, 1);
                double area_r = cvGetCentralMoment(moments_r, 0, 0);

                 // Holding the last and current ball positions //// Prolly don't need this
                static int posX_r = 0;
                static int posY_r = 0;

                int lastX_r = posX_r;
                int lastY_r = posY_r;

                posX_r = moment10_r/area_r;
                posY_r = moment01_r/area_r;

                 // Print it out for debugging purposes
                printf("position_r (%d,%d)\n", posX_r, posY_r);

                 // We want to draw a line only if its a valid position
                if(lastX_r>0 && lastY_r>0 && posX_r>0 && posY_r>0)
                {
                        Com.Write("s");
                        // Draw a yellow line from the previous point to the current point
                       cvLine(imgScribble, cvPoint(posX_r, posY_r), cvPoint(lastX_r, lastY_r), cvScalar(255,0,0),
3);
                }
                 if (posX_r>=0&&posY_r>=0)
                 {
                        if(posX_r>165)
                        {
                                Com.Write("r");
                        }
                        else
                        {
                                Com.Write("l");
                        }
                 }
                 // Add the scribbling image and the frame...
                cvAdd(frame, imgScribble, frame);


                cvShowImage("Threshed", imgRedThresh);

                cvShowImage("Video", frame);
```

```
            // Wait for a keypress
            int c = cvWaitKey(10);
            if(c!=-1)
            {
                // If pressed, break out of the loop
                break;
            }


            // Release the thresholded image+moments... we need no memory leaks.. please
            //cvReleaseImage(&frame);
            cvReleaseImage(&imgScribble);

            cvReleaseImage(&imgRedThresh);

            delete moments_r;
        }

        // We're done using the camera. Other applications can now use it
        cvReleaseImage(&frame);
        cvReleaseCapture(&capture);
        return 0;
}
```