Sensor for Valet-bot

A discussion on the CMU cam as it was used in the Valet-bot project.

By David Wynacht

University of Florida
Electrical and Computer Engineering
IMDL, EEL 5666C
Summer 2003

Introduction

The CMU cam is a camera module that has a image processing unit, camera, power regulator, and serial communication port. My goal was to use the CMU cam to identify a color which would indicate a vacant parking spot.

Summarized Specs

Purchased from Seattle Robotics for $109 assembled.
TTL or RS232 serial communication.
On board voltage regulator with DC input of 6~9V.
Designed by Carnegie Mellon University.
Used at 115k baud, TTL.
RGB or YCrCb data types.
Several data formats available.
ASCII data is transmitted over the serial link.
Palm sized module.
Runs at 17fps.

Integrating the Sensor

This must accomplish several key hurtles which will bring the robot to completing it's ultimate task. First the camera must be able to talk to the Atmel chip. Next the camera must be able to look left and right. Lastly, the camera must be able to see clearly from the robot. The first goal will use the serial communications port on the camera and the second communications port on the Atmel chip. For the next goal the camera will be mounted on a servo so that the camera can be rotated. The last goal will place the camera on the top of the robot or the roof of the vehicle. Any integration of a CMU cam should be concerned with these key issues: Mounting, connection, view, and lighting. If at all possible the best CMU cam integration will provide additional illumination to reduce the sensitivity to it's environment.

Data

The data is exchanged via a serial connection. Both the commands and the data are exchanged in the for of ASCII values. This will require some conversion of the values into integers so that the data can be manipulated. The main data returned from the camera is in the form of what is called the M packet. The M packet data format is as follows:

M mx my x1 y1 x2 y2 #pixels confidence

The M designates which packet type it is. The mx and my are the coordinates of the middle mass of the color blob. The x1 y1 x2 y2 define the box where the color is located. The number of pixels show how many of the pixels in the box are actually the tracked color. Lastly the confidence is an indicator that the tracked object is indeed what you are looking for.

The most important part of a sensor is it's data and that is way I have chosen to use interrupts for receiving. This will make sure that I don't miss any of that important data.

Code

The first program here is one that allows the camera to be controlled by a terminal program or the software that comes with the camera, all while the camera is still connected in the system.

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
SIGNAL(SIG_UART1_RECV)
{UDR0=UDR1;}
SIGNAL(SIG_UART0_RECV)
{UDR1=UDR0;}

void main(void){

        UBRR0L=0x08;
        UCSR0B=0x98;
        UCSR0C=0x06;
        UBRR1L=0x08;
        UCSR1B=0x98;
        UCSR1C=0x06;
        sei();
}
```

The next code is what I used to receive and parse the data:

```
IGNAL(SIG_UART1_RECV)
{
        camdata=UDR1;
        if(pack==0)
        {
                if(camdata=='M')
                {
                        pack=1;
                        indx=29;
                        int j;
                        for(j=0;j<=30;j++){
                        camstr[j]=0;}
                        camstr[indx--]=camdata;
                }
        }
        else
        {
                if(camdata=='\r')
                {
                        pack=0;
                }
                else
                {
                        camstr[indx--]=camdata;
                }
        }

        sei();
}
void cmucomm(char *s)
 {
         while (*s){
         UDR1=*s++;
         servo_delay(400);}
        }
void init_cmucam(void)
{       UBRR1L=0x08;
        UCSR1B=0x98;
        UCSR1C=0x06;
        cmucomm("\rCR 19 32\r");//Turns off the auto gain.
        cmucomm("\rTW\r");
        //cmucomm("\rTC 80 170 90 170 90 170\r");  //Looks for either blue or white
can't remember which.
        }


void main(void){
        lcd_set_ddr();
        lcd_init();
        servo_setcam(80);
        servo_delay(2000);
        servo_setcam(0);
        servo_delay(2000);
        servo_setcam(-80);
        lcd_send_str("Init . . ");
        init_cmucam();
        servo_init();
        lcd_send_str("Ready.");
        int odd=0;
        sei();
while(1)
{       int f=0;


        int l=27;
        int x=0;
        while(camstr[l]!=' ' && l>=0)
        {data0[x++]=camstr[l--];}
        if (x==1) {mx= (data0[0]-48);}
        if (x==2) {mx= (((data0[0]-48)*10)+(data0[1]-48));}
        if (x==3) {mx= (((data0[0]-48)*100)+((data0[1]-48)*10)+(data0[2]-48));}
        x=0;
        l=l-1;
        while(camstr[l]!=' '&& l>=0)
        {data1[x++]=camstr[l--];}
        //if (x==1) {my= (data1[0]-48);}
        //if (x==2) {my= (((data1[0]-48)*10)+(data1[1]-48));}
        //if (x==3) {my= (((data1[0]-48)*100)+((data1[1]-48)*10)+(data1[2]-48));}
        x=0;
        l=l-1;
        while(camstr[l]!=' '&& l>=0)
        {data2[x++]=camstr[l--];}
        //if (x==1) {x1= (data2[0]-48);}
        //if (x==2) {x1= (((data2[0]-48)*10)+(data2[1]-48));}
        //if (x==3) {x1= (((data2[0]-48)*100)+((data2[1]-48)*10)+(data2[2]-48));}
        x=0;
        l=l-1;
        while(camstr[l]!=' '&& l>=0)
        {data3[x++]=camstr[l--];}
        //if (x==1) {y1= (data3[0]-48);}
        //if (x==2) {y1= (((data3[0]-48)*10)+(data3[1]-48));}
        //if (x==3) {y1= (((data3[0]-48)*100)+((data3[1]-48)*10)+(data3[2]-48));}
        x=0;
        l=l-1;
        while(camstr[l]!=' '&& l>=0)
        {data4[x++]=camstr[l--];}
        //if (x==1) {x2= (data4[0]-48);}
        //if (x==2) {x2= (((data4[0]-48)*10)+(data4[1]-48));}
        //if (x==3) {x2= (((data4[0]-48)*100)+((data4[1]-48)*10)+(data4[2]-48));}
```

```
x=0;
l=l-1;
while(camstr[l]!=' '&& l>=0)
{data5[x++]=camstr[l--];}
//if (x==1) {y2= (data5[0]-48);}
//if (x==2) {y2= (((data5[0]-48)*10)+(data5[1]-48));}
//if (x==3) {y2= (((data5[0]-48)*100)+((data5[1]-48)*10)+(data5[2]-48));}
x=0;
l=l-1;
while(camstr[l]!=' '&& l>=0)
{data6[x++]=camstr[l--];}
if (x==1) {pix= (data6[0]-48);}
if (x==2) {pix= (((data6[0]-48)*10)+(data6[1]-48));}
if (x==3) {pix= (((data6[0]-48)*100)+((data6[1]-48)*10)+(data6[2]-48));}

//x=0;
//l=l-1;
//while(camstr[l]!=' '&& l>=0)
//{data7[x++]=camstr[l--];}
//if (x==1) {conf= (data7[0]-48);}
//if (x==2) {conf= (((data7[0]-48)*10)+(data7[1]-48));}
//if (x==3) {conf= (((data7[0]-48)*100)+((data7[1]-48)*10)+(data7[2]-48));}
```

Assessment

While in testing that CMU cam perform quite well, when it came to the demonstration the CMU cam always had a problem. I did notice that the processor in the module was usually warm. On my robot the camera is in an enclosure that attaches to the servo. The combination of the process's heat and the voltage regulator's lead to the device overheating. With most electronic devices temperature is a big issue. It is know that CCD camera are especially affected by heat. I don't know if a CMOS camera has different heat issues or not, but this module does. After, letting the device cool down it again processed color recognition. The cmu camera need a very tightly controlled environment. If the environment is going to change then, the most success of the device would be using the TW command. This is the track window command, meaning that the color to track is held in front of the camera for a length of time. This way if the light has changed from different location and t eh camera is not seeing the same color, this will allow the device to look at the color in this new illumination and temperature.

In essence if the right amount of patience and knowledge of the device is in control of how the camera is utilized then you can achieve what you need it to do. This is not a easy to use device nor is it a reliable device by far. If I were to create a full-scale version of my robot I would use a more controllable and reliable system. Such as, standard protocol devices, like USB cameras or

FireWire device, and a computer as the vision processing core. As with anything as you increase to quality your output increases in quality.