Special Sensor Report:

CMUcam

University of Florida
Department of Electrical and Computer Engineering
EEL5666
Intelligent Machines Design Laboratory

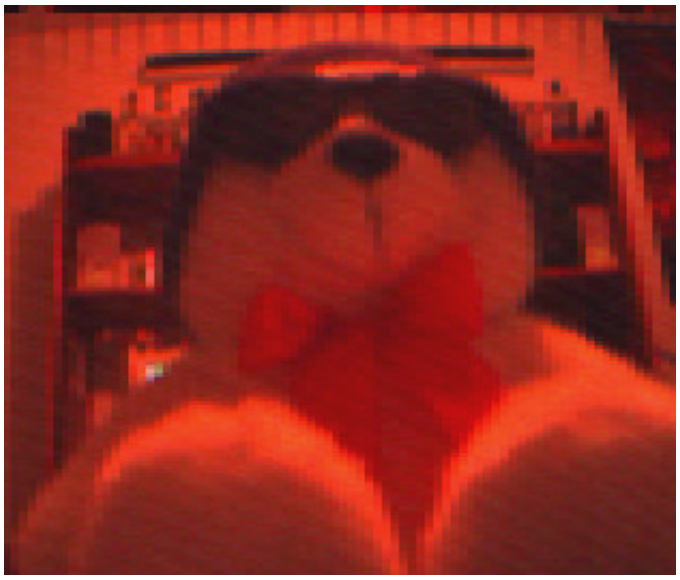
Brian Ruck
April 22, 2003
Instructor: Dr. Arroyo
TAs: Uriel Rodriguez
Jason Plew

Introduction

Nigel uses the CMUcam to find people wearing the color blue or have blue somewhere on there person. The camera, made by Carnegie Mellon University, was purchased at Seattle robotics, (www.seattlerobtics.com) for $109.00, pre-assembled. Acroname also sells the cam but it doesn't come pre-assembled. After looking at a classmate's unassembled CMUcam from Acroname it is my personal recommendation to just get it pre-assembled. The cam from Seattle robotics also comes with a built in IR-filter, which proves useful when tracking colors when outside light is emitting through.

CMUcam

The CMUcam's abilities include, position and size tracking of colorful or bright objects, measure the RGB and YUV characteristics of an image region, display images and bitmap of objects via serial port. For Nigel's purpose, the use of the CMUcam is quite simple, Nigel will utilize the color tracking feature and interface with the board to control its servos to approach a person wearing the color blue. The images "dumped" by the camera had a very dark red tint thus discoloring objects placed in front of it. A picture of a stuffed bear displaying a variety of colors is shown below to demonstrate the images on the cam.



The bear is white, with a red bowtie around its neck, black glasses and a blue hat. This picture was taken under bright lighting conditions in my room. As you can see, this type of lighting is not good and tracking a color like blue would prove to be difficult. The best images were obtained when I was under fluorescent lighting, colors were more distinct and the picture was much clearer.

Camera Integration

The CMUcam itself uses a SX28 microcontroller that communicates either through a RS-232 or a TTL serial port. To integrate with my AVR-MEGA progressive development board I communicated through the RS-232 and level shifted serial port. Communication through this port uses the following parameters:

- 115,200 baud
- 8 data bits
- 1 Stop bit
- No Parity
- No Flow control

These parameters are ideal for communication with a computer, i.e. hyper terminal. However, one can interface it with a board just the same. The camera has several commands that are rather easy to use; the command that was relevant for Nigel was the track color command or "TC". By using the CMUcam software that came with the cam,

I was able to easily dump a frame, get color coordinates and obtain values for the minimum and maximum intensities of the camera's main colors red, blue, and green. Once those values are entered, the track color function can be assessed and the once the colored object is placed in front of the cam the green light will blink indicating that the color being tracked is found, what is being returned from the camera is an M-type packet which consists of:

M mx my x1 y1 x2 y2 pixels confidence

This string is what I will use to calibrate Nigel to chase the color blue.

M=represents M type packet

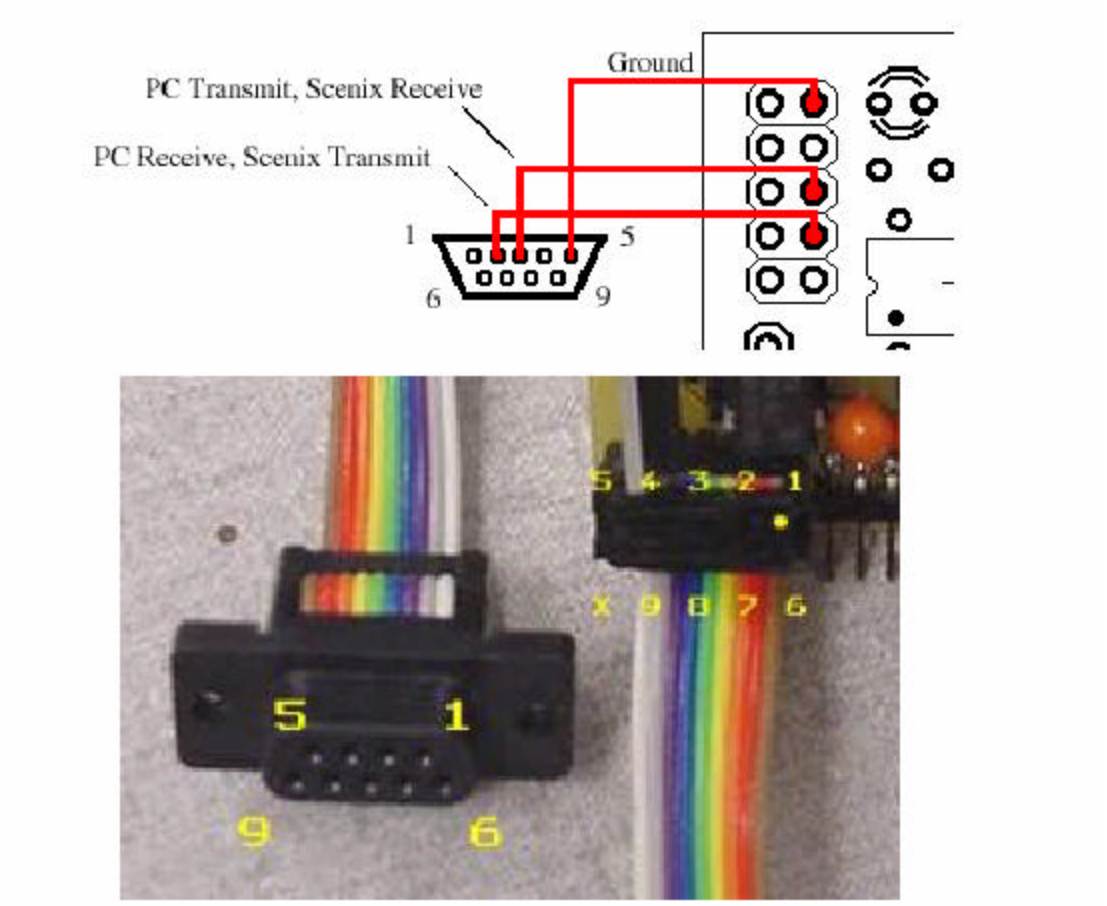
mx=middle mass x

my=middle mass y

x1, y1, x2, y2 represent the coordinates of the box that the camera sees when it is tracking.

For Nigels purposes, I am only concerned with the middle mass x (mx) where he will turn left, right or go straight depending on where the color goes.

This method of obtaining "min and max" RGB color coordinates was extremely easy, but communicating with the board proved to be rather tedious. The problem was that I was able to send commands to the camera but not able to receive packets of info back. After much frustration and collaboration, a method of communication was established. Before communication between the cam and the board can be established it is necessary to figure out where the Tx, Rx pins are located on the cam. As shown below:



To receive packets from the camera, the Tx and Rx pins on the cam must be crossed to connect properly to the Rx and Tx pins of the board. In other words, pins 2 and 3 from the camera need to be crossed so the transfer (pin2) of the cam is the receive pine of the board (pin 3) and the receive pin of the cam (pin3) is connected to the transfer of the board (pin 2). By doing this and cutting all the other wires we were able to establish a clean and steady communication system. Thus, I was able to send a string of commands and receive the middle x packet byte and display it on PortC. The following code was originally written by Kyle Tripician (see Aluminator/Spring2003) and modified to fit Nigel's tracking coordinates for the color blue.

```
//C code used to test the track color command and get middle x value on PortC.
/*
Brian Ruck
CMUcam code
```

```
Courtesy of
Kyle Tripician
*/

#include <io.h>
#include <sig-avr.h>
#include <stdlib.h>
#include <interrupt.h>
#include <progmem.h>
typedef unsigned char u08;
typedef unsigned int u8;
typedef           char s08;
typedef unsigned short u16;
typedef           short s16;
#include "uart.h"
#include "delay.h"
volatile u8 temp;
volatile u8 i=0;
volatile u8 cmudat[9];


SIGNAL(SIG_UART_RECV){
        temp=inp(UDR);
        if(temp != 0x3A){
                if(temp == 0x20 || i==9){
                i=0;
                }
                else if(i==0){
                        if(temp == 0xFF){
                                cmudat[i]=temp;
                                i++;
                        }
                }
                else if(temp !=0x20 && i<9){
                cmudat[i]=temp;
                i++;
                }
        }

}
void blink(void){
        uartstring("L1 1\r");
        delay(1500);
        delay(1500);
        delay(15000);
        uartstring("L1 0\r");
```

```
        delay(15000);
        delay(1500);
        delay(1500);

        }
int main(void){
        outp(0xFF,DDRC);
        delay(10000);
        uartinit();
        delay(10000);
        uartstring("PM 1\r");
        delay(1000);
        uartstring("RM 3\r");
        delay(1000);
        sei();
        delay(1000);

        while(1){
        uartstring("TC 42 62 33 53 40 90\r");
        outp(cmudat[2],PORTC);
        delay(20000);
//      blink();
        }
}
```