03-23-06

Andrew May

Adam Barnett and Sara Keen

A. A. Arroyo

University of Florida

Department of Electrical and Computer Engineering

EEL 5666

Intelligent Machines Design Lab

Special Sensor Report
Special Sensor: Carnegie Mellon University Camera (CMUCam)

Ahead:
Functionality
Experimentation
Conclusions

Functionality

I have various problems with this camera. The first is the power switch. This "switch" is actually a sharp clip that is smaller than 0.5" X 0.2". I soldered a wire in its place in the "ON" position. The next issue was lighting. The CMUCam has an excellent auto-white adjust. However, in incandescent bulb lighting, the colors blur into one indistinguishable mass.

I took reads from the side of the NEB rotunda to see if the camera would not work there.

This environment is NOT the same as Media Day because there is more light here than in the center of NEB. I will have to try again in the NEB rotunda. Here, the light is fluorescent – the best kind! One problem with this experiment is the results are from movement over the black and beige tiles (NOT the white paper that I want to use for my demonstration!!!)

Ambient color means were (Red_min Red_max Green_min Green_max Blue_min Blue_max):

130 170 90 155 47 93

Tracking worked very well (confidence values were high).

Picture quality was excellent.

The CMUCam was facing downward 30 degrees from horizontal. I will need another trial on top of white glossy construction paper with a, solid colored object.

The primary method of interfacing the CMUCam is serially. An RS-232 connection to the camera's DB9 connector is all the hardware necessary. In the software, the UART is configured to accommodate the camera. After that, the CMUCam is initialized. Lastly, the track color, or TC, command is executed. TC returns a set of values that say where the middle of the color it was given and the accuracy is.

Experimentation

I pointed my CMUCam in one direction. With a slight change in height the Red, Green, and Blue (RGB) intensity values changed dramatically. Explicitly, every 0.5" caused the following changes: R 10, G 8, and B 4.

I placed white construction paper onto a table in the corner of the room where I will be demonstrating my robot if I demonstrate it. At base height of 4" from the table approximately, the RGB values were about 101 111 56.

However, on the floor in the center of the said room, RGB values became 101 121 64.
These last two data came from pointing the camera downward at a 45 degree angle. When I pointed the camera straight the ambient light was seen as a RGB value of 130 158 80.

My next step in this experiment was to see how my camera behaved as it was, tilted down at a 45 degree angle without any LEDs pointing at the subject. With the same white construction paper on the same place on the floor, I scanned a red object (A strawberry Nutrigrain bar). The resulting RGB value was 113 123 60. This shows a lower intensity of light of all colors and more blue than red. This means the package color was not seen because the package is red. I infer this RGB value means that less light is getting to the camera than necessary to record the color that my eyes can see. So, to the camera, the red package looks more like a black one. Consequently, I will need to add white LEDs to illuminate my camera's subject.

I repeated the previous step with a blue object (A blueberry Nutrigrain bar). The RGB value recorded was 95 120 70. Again the RGB values went down but in a different way.

So, I tried looking straight at the packages instead. Now, the red object had an RGB value of 128 100 53. The blue object had 90 100 80. The blue data shows that green and red is more intense than red. The package is blue, though. As a result, the actual colors never were seen by the camera. This no-color identification could be from the reflective packaging. I propose testing construction paper. If construction paper is read correctly color-wise, then I might glue paper around the Nutrigrain bars.

Still, the reflective packaging generates enough of a difference between the RGB values of the red object and the blue object to recognize them in software through the track color command on the CMUCam. This differentiation came from more light that bounced off the colored package and into the camera. Hence, I feel that the addition of white LEDs will allow me to obtain the contrast between colors that I need in order to differentiate between colors robotically.

To test the get mean function, which yielded the RGB values, I shot the colors straight and without LEDs to approach the final environment caused by the addition of LEDs at a downward 45 degree angle.

Fig. 1

Procedure: I put a red object up as shown in Fig 1. Then, I measured the distance from the red object, but I started at a low distance until I saw more Red value than any others. Then, I moved back and recorded the following values every 0.5” starting at 1”, where the distance is about from the edge of the widest side of the red object to the closest point on the camera. The data is in tables and charts. The tables have the distance in inches, red mean, green mean, and blue mean.

Distance in inches	Red	Green	Blue
1	150	78	48
1.5	142	67	39
2	144	76	44
2.5	138	87	51
3	137	94	54
3.5	135	100	56
4	135	107	58
4.5	132	108	58
5	134	113	62
5.5	132	110	57
6	136	120	64
6.5	132	120	63
7	131	119	62

Table 1

Chart 1

Distance in inches	Red	Green	Blue
1	146	72	45
1.5	140	67	39
2	147	73	39
2.5	135	84	46
3	136	91	47
3.5	132	99	51
4	133	102	51
4.5	133	108	54
5	143	121	61
5.5	136	116	55
6	135	117	57
6.5	136	120	58
7	131	116	55

Table 2

Chart 2

Distance in inches	Red	Green	Blue
1	86	92	72
1.5	79	82	77
2	80	85	70
2.5	94	101	68
3	91	98	68
3.5	95	104	66
4	102	112	68
4.5	101	112	66
5	109	119	69
5.5	111	122	71
6	103	117	64
6.5	109	122	68
7	91	108	56

Table 3

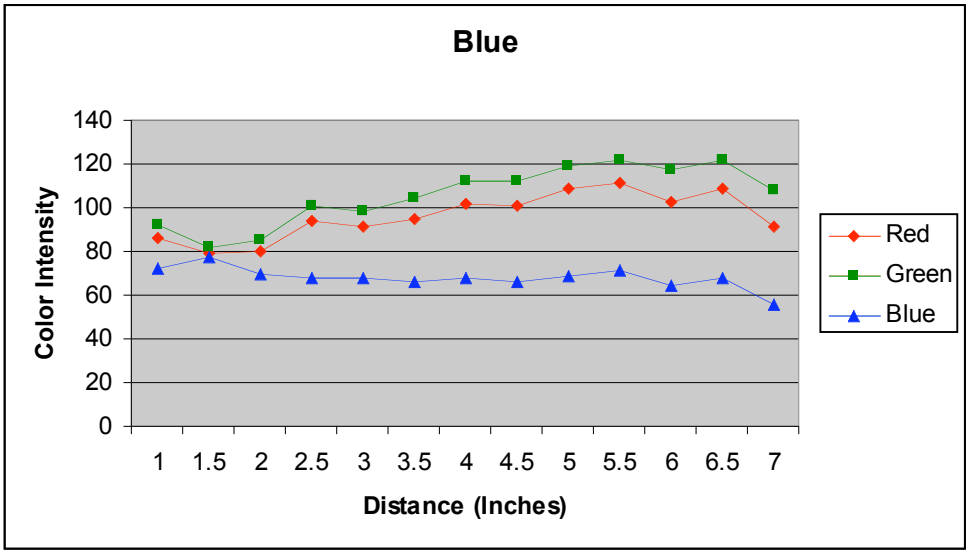

Chart 3

The raw data for the previous tables and charts:

Red

Trial 1
1 150 78 48
1.5 142 67 39
2 144 76 44
2.5 138 87 51
3 137 94 54 (red drops but blue and green rises because the LIGHT hitting the camera goes from reflected red primarily to white)
3.5 135 100 56
4 135 107 58
4.5 132 108 58 (this is the threshold distance because the color values is almost constant beyond this distance)
5 134 113 62
5.5 132 110 57
6 136 120 64
6.5 132 120 63
7 131 119 62

Trial 2
1 146 72 45
1.5 140 67 39
2 147 73 39
2.5 135 84 46
3 136 91 47
I knocked down the set up I had here…
3.5 132 99 51

4 133 102 51
4.5 133 108 54
5 143 121 61
5.5 136 116 55
6 135 117 57
6.5 136 120 58
7 131 116 55

Trial 3
1 129 79 42
1.5 147 71 39
2 138 71 39
2.5
…aborted because the measurement tape was off

I repeated this previous color mean experiment with the blue object once only.

Blue

1 86 92 72
1.5 79 82 77
2 80 85 70
2.5 94 101 68
3 91 98 68
3.5 95 104 66
4 102 112 68
4.5 101 112 66
5 109 119 69
5.5 111 122 71
6 103 117 64
6.5 109 122 68
7 91 108 56

Conclusions

As true as these data sets are, I still have some issues to deal with. First, there are differences between how I will demo my robot's CMUCam and how I tested my robot's CMUCam. Namely, I didn't use white LEDs, I didn't have the CMUCam at the same height and angle that it will be in finally, and I didn't try the CMUCam for a couple complete trials on the table that I may include to show off my robot's edge of the world detection if I need to. Next, there are issues with the power. I gave the CMUCam about 5.9 V. The light was on and the serial data was obtained. However, the manual suggests that 6 – 9 V run the CMUCam instead. Lastly, I have the track color (TC) command to test. This command shows where the center of the color desired is in the camera window frame. This function allows the user to adjust the camera until it points at the center of the color, for example.