A Vehicle Implementation of a Color Following System using the CMUcam3

J.D. Bikman
UNC Charlotte
jbikman@uncc.edu

T.W. Meiswinkel
UNC Charlotte
twmeiswi@uncc.edu

J.M. Conrad
UNC Charlotte
jmconrad@uncc.edu

Abstract

By creating a center of mass for a specific color, color tracking, and controlling servos relative to the distance from the center of mass, the *CMUcam3 and Gears SMP robotics platform can be used to build a color tracking system to autonomously track a certain color within the Y'CrCb color range. Using pulse-width modulation arrays to control the servomechanic DC motors, a control system based on mean image data can be implemented to steer and direct an autonomous CMUcam3 driven vehicle platform.*

1. Introduction

Graduate and undergraduate students in the UNC Charlotte Electrical and Computer Engineering department have developed a color tracking system for an autonomous vehicle. The goal of this project was to test the capabilities of the CMUcam3 (CC3) system (Figure 1) for possible future use as a sensor in a munitions clearance project. This was not the end result of the CC3 project; enough work was done with the CC3 however to develop a configurable color following system with uses in future applications involving servo motor control [1].

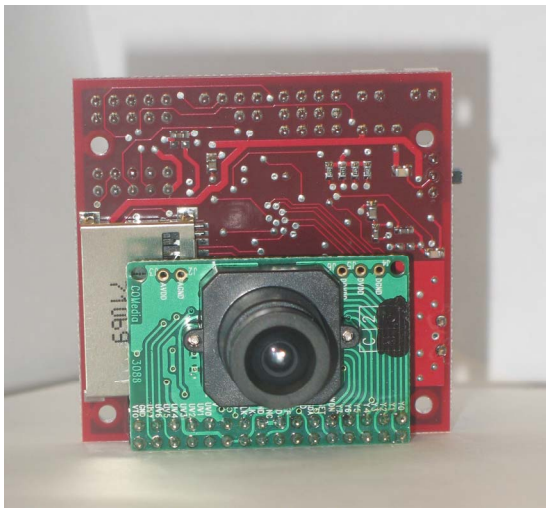

Figure 1: CMUcam3 hardware

Color is an important indicator for environmental differentiation between objects. This use of color does not require the use of computationally heavy edge detection algorithms. Using machine vision for object detection allows for object profiling and targeting, which is not possible using more crude means of object detection such as SONAR. Differentiation between objects on the basis of color has been done in the past using Newton Lab's Cognachrome system [2]; however, this system costs between 2500 and 5500 dollars per unit, depending on the buyer. The CMUcam3 system used for this project costs around 250 dollars per unit, yielding much less overhead.

2. Camera Device and Color Spaces

A Complementary Metal Oxide Semi-conductor (CMOS) camera sensor such as that used in the CC3's OV6620 camera produces a discrete voltage value for each pixel. This contrasts the operation of a Charged Coupled Device, which operates as an analog shift register, translating directly one photon of light to one pixel of data. CMOS camera sensors consume much less power than CCD sensors, which is beneficial for an embedded systems application, but the CMOS image is more susceptible to noise [3].

A black-level calibration algorithm from [4] is applied to the CMOS sensor in the OV6620 and provides a normalized digital image for later processing. An analog video port is provided by the CMUcam3 which outputs in the YUV color space. The yellow channels are identical in both Y'CrCb and in YUV; the UV channels in the YUV format as well as the CrCb channels in the Y'CrCb format are used for saturation and hue chromaticity, and provide red and blue channel differencing [5]. The conversions from RGB to Y'CrCb color space are:

"Y = 0.59G + 0.31R + 0.11B",
"Cr = 0.713 x (R - Y)", and
"Cb = 0.564 x (B - Y)."

The formulas for converting RGB into the YUV color space on the OV6620 are [4]:

"Y = 0.59G + 0.31R + 0.11B",
"U = R-Y", and
"V = B-Y"

3. Platform Chosen

The CMUcam3 system (Figure 1) consists of a CMOS camera with a MAX232 serial transfer port for flashing programs, SD card port for data storage, a Pulse-Width Modulated (PWM) servo motor controller, an analog PAL output in the Y'CrCb color space, as well as a robust, fully

open-source API for programming the device. A portable Lilliput 233GL-25NP 2.5 inch liquid crystal screen [6] was connected as an external display for calibration and debugging purposes. The Gears SMP platform (Figure 2) was connected to the PWM ports on the CMUcam3, which allowed direct control of the pair of skid-steered DC motors. The unified control system of the PWM ports through the API allowed the design to be focused on the color tracking algorithm [7].

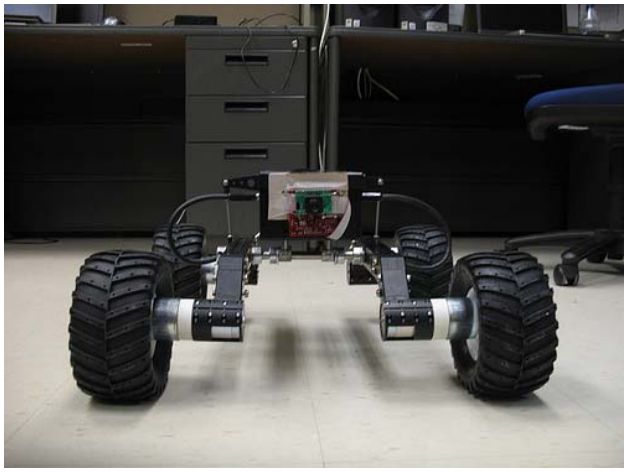

Figure 2: Vehicle platform with CMUcam3

Writing programs for the CC3 on the Ubuntu 8.04 Linux operating system involved setting up an environment to allow cross-compilation for the ARM Embedded Application Binary Interface (EABI) and configuring this cross-compiler for use with the GNU gcc compiler by adding path variables for both the ARM EABI compiler and the lpc21isp_unix installer to the .bashrc file. An undocumented, though necessary, step for allowing use of the ARM EABI compiler on Ubuntu 8.04 was the simple command "apt-get install build-essential", which gathered necessary backbone compiler tools from the apt-get server.

Connecting to the CC3 by means of a terminal required use of the serial interfacing program Minicom, with default baud rate set to 115200 8N1. Installing programs to the CC3 required using the lpc21isp_unix installer.

A sample command used to install a program to the CC3 using the lpc21isp_unix installer is:

```
./lpc21isp cmucam2_lpc2106-cmucam3.hex
        /dev/ttyUSB0 115200 14746
```

While connected to the CC3 via Minicom, information could be output as desired to the program by use of the "printf()" function in C. More information regarding the interface of the CMUcam3 is located at [8].

4. System Algorithm

4.1 Color Tracking Theory

Capturing an image on the CC3 and outputting the video feed in real-time requires setting the camera to view within the Y'CrCb color space at high resolution. In order to create the proper target size for the color tracked in this application, shrinking the window to a region of interest of size $1/4^{th}$ of that of the original image is necessary. After shrinking to this smaller window, the mean color of the image can be obtained by calling the function "simple_get_mean()" to gather the mean color of the image as illustrated in Equation 1 below:

$$mean\_color = \frac{1}{i \times j} \sum_{i=1}^{cols} \sum_{j=1}^{rows} pix(i, j) \quad (1)$$

This value of *mean_color* is then added to a variable denoted s_pkt of the API class cc3_color_info_pkt_t; the information from this initial image is used to set maximum and minimum values for each color channel. After this initialization, the region of interest is then reset to the original image size. This equation is similar to the approach by Lazebnik, et.al. [9], except for the fact that it only tracks one color.

A threshold range is determined in which every pixel in the image is compared iteratively by scanning through the image and finding all the pixels that fall within a given threshold range. A target's *centroid* values represent the coordinates of its center within the image frame. Cumulative centroids for the x and y dimensions are created from the average number of "good pixels" (pixels of color interest). This accumulation takes place in the API function called "simple_color_scanline." Values for pixel density are then created by dividing the number of "good pixels" by the total number of pixels. This is done by gathering cumulative data and sorting this data in the function called "cc3_track_color_scanline_finish." The total number of pixels in this case was 101,376 due to the resolution, 352x288.

Also in this function, the true centroid values for x and y are derived from their respective cumulative x and y centroid values computed from the previous function. The true centroid values are found by dividing each respective cumulative centroid value by the number of pixels in the image.

4.2 Motor Driving

Comparing the obtained centroid values with values of x_mid and y_mid (which respectively represent half the image's width and height, respectively), the statistical offset of the centroid from the center of the image may be calculated and used inside of conditional statements for

controlling the servos. In the following code segment, the condition for a centroid is illustrated 20 pixels to the right of the center of the image frame. The variable t_pkt represents an API class variable of type cc3_track_pkt_t.

```
while(t_pkt.centroid_x >= (x_mid + 20))
```

The following commands are sent via the gpio pins to control the motors and turn the platform to the right:

```
cc3_gpio_set_servo_position
   (0, SERVO_REV_SPD[2]);
cc3_gpio_set_servo_position
   (2, SERVO_FWD_SPD[2]);
```

The line `cc3_gpio_set_servo_position (0, SERVO_REV_SPD[2])` tells the left skid servos to move backwards at the speed assigned to the third assigned value in the SERVO_REV_SPD array. The servos used in the project were endlessly rotating DC motor servos, operating with PWM. If the PWM signal sent to the servo was a significant number less than 127, the servo would drive in reverse. If the signal sent to the servo was a significant number greater than 127, the servo would go forward. If the signal sent to the servo was between approximately 115 and 135, the servo would not move in either direction.

Because of the while loop continuously sending out PWM signals to turn left, conditions must be included inside of the while loop after the "turn left" command to compensate for this over-steering.

```
if((t_pkt.centroid_x >= (x_mid - 10)) &&
(t_pkt.centroid_x <= (x_mid + 10)))
{
   cc3_gpio_set_servo_position
      (0, SERVO_FWD_SPD[2]);
   cc3_gpio_set_servo_position
      (2, SERVO_FWD_SPD[2]);
}
```

This code snippet shows a condition for when the centroid falls in-between the middle 20 pixels of the image, with respect to the x-axis. When this takes place, the robot drives forward until the next condition is met:

```
if((((t_pkt.x1 - t_pkt.x0) * (t_pkt.y1 -
t_pkt.y0)) - t_pkt.num_pixels) > 80000)
{
   cc3_gpio_set_servo_position
      (0, SERVO_STP);
   cc3_gpio_set_servo_position
      (2, SERVO_STP);
}
```

This condition shows the centroid bounding box being used as a condition to prevent the platform from driving forward when the difference between the centroid and the total number of pixels exceeds 80000. This prevents the unit from driving into the target when it is too close. It is important to remember to call the function simple_track_color(&t_pkt); at the beginning of each loop to keep the data fresh and updated.

5. Implementation Results

The CC3 camera system and driver was tested in the lab by observing motor control signals in response to moving a green-colored target in front of a camera.

The CC3 camera system was then connected to the Gears EDS Platform and its motor drivers. The green-colored target was mounted onto the back of another Gears EDS Platform. This second vehicle's motor drivers were controlled with a model aircraft controller, as shown in Figure 3. The objective was to have the CC3-based vehicle follow the remote-controlled vehicle. We were successful for several seconds at a time.

While testing the CC3 system, it was noticed that the reflexes of the camera were rather jerky. The camera would oscillate while the target remained at rest, which may be the result of the speed of the PWM signals sent to the motors. In other autonomous robotics applications, the DC motor controller functionality is often delegated to a peripheral dedicated circuit such as an H-bridge. This is done to dampen the signal sent to the servos and allows for smoother operation. In future implementations of the CC3 system, use of an H-bridge or something similar would be advised. Testing found an optimal distance between the CC3 and the color target of about 1-3 meters.

Figure 3: Example of autonomous vehicle with CMUcam3 following a remotely controlled vehicle with a colored card.

6. Conclusion and Future Implementations

The CC3 system developed over the course of this project can be improved by using better algorithms by using training data to provide a statistical model for images. Color tracking has many applications resultant from its

quick object detection; general applications including swarming, object recognition, and object detection. Because of the low cost and open sourced architecture of the CMUcam3, the same capabilities of the Cognachrome system may be developed at a much lower cost.

Immediate future work will be the effort to ensure the CC3 vehicle system reliably follows the color target. The tracking performance improves as more is learned about the capabilities of the camera. An articulating pan and tilt turret camera base for example could used to improve the sensor input range of the CC3.

Currently in development for the CC3 color tracking system is the autonomous following and tracking of a line of Gears EDS platform vehicles, with one remotely-controlled vehicle at the front. This will be the proof of concept for a swarming robotics system.

An improved version of the CC3 color tracking system could include multiple centroid bounding regions for multiple targeting. Also, an improvement to the CC3 control system could base actions on looking up histogram based lighting/color data. An autonomous vacuuming robot such as the Roomba could have the owner take pictures of the bathroom floor and the carpet, put these pictures onto an SD card and put the SD card into the CC3's SD card reader. Using the color and lighting data from these pictures, different behaviors for different environments could take place [10]. In the bathroom for example, the CC3 would know that the floor has white tiles and would likewise use a household cleaner rather than just the vacuum cleaner. On carpet, the CC3 would know just to deploy a vacuum and not spray a household cleaner.

7. References

[1] M.J., Zapata, W.J. Haynes, N. Kannen, M. Sullivan, and J.M. Conrad, "An Autonomous Vehicle for Space Exploration" *Proceedings of IEEE SoutheastCon 2008*, pp.15-20, April 2008.

[2] Newton Research Labs, "The Cognachrome Vision System," http://www.newtonlabs.com/cognachrome/

[3] Dave Litwiller, "CMOS vs. CCD: Maturing Technologies, Maturing Markets," *Photonics Spectra*, pp. 54-59, August 2005.

[4] Omnivision Technologies, Inc., "OV6620 Single-Chip CMOS CIF Color Digital Camera Datasheet," Version 1.4, 13 May 2000.

[5] A. Ford and A. Roberts, "Color Space Conversions," August 11, 1998, available at URL: http://www.poynton.com/PDFs/coloureq.pdf

[6] Lilliput, Inc., "Model Number: 233GL-25NP Technical Specifications," available at URL: http://www.lilliputweb.net/np233.html

[7] A. Rowe, A.G. Goode, D. Goel, and I. Nourbakhsh, "CMUcam3: An Open Programmable Embedded Vision Sensor", tech. report CMU-RI-TR-07-13, Robotics Institute, Carnegie Mellon University, May, 2007.

[8] Carnegie Mellon University, CMUcam3 Data Sheet," Version 1.02, September 22, 2007, available at URL: http://www.cmucam.org/wiki/Documentation

[9] H. Schneiderman, "Feature-Centric Evaluation for Efficient Cascaded Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-8, June 2004.

[10] S Lazebnik,. C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,", *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol.2, null.*, pp. 2169-2178, June 2006.