A Low Cost Embedded Color Vision System

Anthony Rowe[1], Charles Rosenberg[2], Illah Nourbakhsh[3]

Carnegie Mellon University, Pittsburgh, PA, USA
[1]*Electrical and Computer Engineering Department, agr@andrew.cmu.edu*
[2]*Computer Science Department, chuck@cs.cmu.edu*
[3]*Robotics Institute, illah@ri.cmu.edu*

Abstract

In this paper we describe a functioning low cost embedded vision system which can perform basic color blob tracking at 16.7 frames per second. This system utilizes a low cost CMOS color camera module and all image data is processed by a high speed, low cost microcontroller. This eliminates the need for a separate frame grabber and high speed host computer typically found in traditional vision systems. The resulting embedded system makes it possible to utilize simple color vision algorithms in applications like small mobile robotics where a traditional vision system would not be practical.

1 Introduction

There are many examples in the literature of simple computer vision algorithms proving to be extremely useful in a variety of applications [2], [4], [5], [7], [12], [14]. However the usefulness of these algorithms is often limited by the cost and complexity of the hardware needed to implement them. Such systems traditionally consist of a camera, a frame grabber, and an associated computer to interface to the frame grabber and execute the algorithm. Recent hardware developments now make it possible to greatly simplify and reduce the cost of these systems. The two developments which we take advantage of in this work are low cost CMOS color camera modules and high speed, low cost microcontrollers. A major advantage of CMOS versus CCD camera technology is the ability to integrate additional circuitry on the same die as the sensor itself. This makes it possible to integrate the analog to digital converters and associated pixel grabbing circuitry so a separate frame grabber is not needed. As microcontrollers have become more prevalent their cost has decreased and their capabilities have increased. This makes it possible to perform simple pixel processing "on the fly" as the pixel values are scanned out of the camera making a full frame buffer unnecessary in many situations. This suggests that it should be possible to team a CMOS camera chip with a low cost microcontroller and implement a simple vision system.[10] We have constructed a functioning system based on this idea which we describe in the remainder of this paper. The fully assembled system is commercially available for a cost of $109.[9]

2 System Details

Our vision system is designed to provide high-level information extracted from a camera image to an external processor that may, for example, control a mobile robot. In a typical scenario, an external processor first configures the vision system's streaming data mode, for instance specifying the tracking mode for a particular bounded set of RGB values. The vision system then processes the data in real time and outputs high-level information to the external consumer. The following sections describe the details of the system which we have implemented.

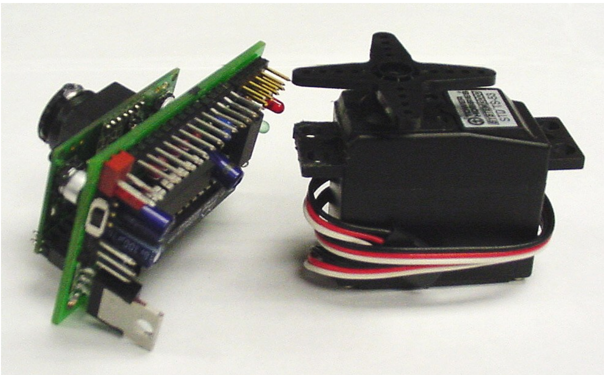

Figure 1: *The microcontroller board mated with the CMOS camera module. A standard size hobby servo is shown for scale.*

2.1 Hardware System

The hardware for our system consists of a three chip design. The first two chips are the OV6620 CMOS camera and the SX28 microcontroller. The third chip is a simple level shifter for the RS232 serial data. To keep the design simple, the data bus, synchronization pins and configuration bus from the OV6620 are directly connected to the SX28 without the aid of any glue logic. The SX28 waits for incoming data to stream from the camera and processes it in real time. It then relays the extracted high level information to the outside world via an asynchronous serial interface implemented in software. The complete vision system is 1.75" $\times$ 2.25" and less than 2" deep with the camera module and lens attached, see Figures 1 and 2. The system operates at 5 volts and draws about 200 milliamperes of current.

The image input to the system is provided by an Omnivision OV6620 CMOS camera on a chip.[8] The CMOS camera is is mounted on a carrier board which includes a 4.9 mm F2.8 lens and a few supporting passive components such as a 17 MHz clock crystal. Different lenses are available for customizing the optics. By itself, the board is free running and will output a stream of 8 bit RGB or YCrCb color pixels along a 8 or 16 bit wide data bus. Synchronization signals, including a pixel clock, are then used to read out data and indicate new frames and horizontal lines. The CMOS image array contains 101,376 pixels and supports resolutions of up to 352 $\times$ 288 with a maximum refresh rate of 60 frames per second.[8] CMOS camera parameters such as color saturation, brightness, contrast, white balance, exposure time, gain and output modes are programmable using a standard serial I^2C interface. To utilize video data from the OV6620 one must properly initialize the camera and then remain synchronized with each of its output signals. An independent monochrome analog output exists that can be used for external monitoring of the image. Due to the nonstandard frame rate utilized in our system a multisync display device is necessary to properly decode the image.

The microcontroller that is used to process the video data is a Ubicom SX28 operating at 75 MHz, model number SX28AC/DP.[13] It is housed in a standard 28 pin narrow DIP package. The SX28 is a RISC processor and operates at 75 MIPS. It has a 2048 word flash programmable EPROM and 136 bytes of SRAM. Although it has few hardware peripherals, it has fast and deterministic interrupts as well as three flexible multi-bit I/O ports that allow software to emulate standard hardware peripherals as "virtual" peripherals. Using these virtual peripherals, we implemented a serial UART port, a standard hobby servo PWM output port and can control a status LED in our system. With our hardware design, it is also possible, using a pass-through PC104 style connector to join multiple SX28 vision boards on a single camera bus. This allows for parallel processing of the image data in what we call slave mode. Using this "slave mode" two microprocessors can be attached to the output of a single CMOS camera, allowing two different image operations to be performed in a fully synchronized fashion.

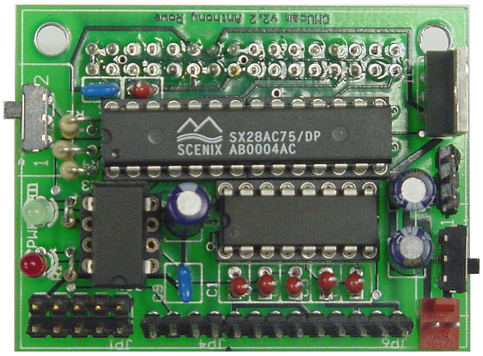

Figure 2: *Detail of the assembled microcontroller board, 1.75" $\times$ 2.25". Visible are the microcontroller at the top, the RS232 level shifter below on the right, and the clock oscillator below on the left.*

2.2 Firmware System

The main challenge that we had to overcome in developing the software for this system was the small amount of RAM available in most microcontrollers. In our case, with only 136 bytes of RAM it is impossible to buffer an entire image. In fact, it is not even possible to buffer an entire row of color image data. To work within such limitations, we were required to process the data as it streams from the camera with little or no buffering. By utilizing the data as it streams from the camera we were able to perform basic image processing during the time between pixels. To reduce processing time we drop the trailing G component of the sensor's RGBG (or CrYCbY) pixel group and skip every other pixel. At the end of each row and once at the end of a frame there is extra time that can be used for additional post processing and transmission of data. This method limits our horizontal resolution to 80 RGB pixels, but does not affect the maximum vertical resolution of 143 pixels. One of the main algorithms that the system is capable of implementing with this method of processing is a simple form of color blob tracking.

All firmware for the vision board was written in C and compiled using the ByteCraft SXC v2.0 compiler. When compiled the current firmware requires 2035 words of ROM and at some points utilizes all but 1 byte of the SX28's RAM. Needless to say the firmware had to be coded very carefully.

Color Blob Tracking. The color blob tracking algorithm allows the user to enter a minimum and maximum bound for each of either the three RGB or YCrCb channel values, depending on how the camera is configured. Each pixel in the buffer is compared against the user specified bounds. The coordinates of the pixels that fall within the color bounds are compared against previously stored coordinates to generate a bounding box. This simple method requires that the SX28 store little global information about the image. The stored data includes the upper left x1, y1 coordinate and the lower right x2, y2 coordinate that enclose pixels which satisfy the color bounds. We also count how many pixels actually fall within the color boundaries. Once the entire frame has been processed, some additional post processing operations are completed. In particular, a scaled ratio between the total sum of pixels within the color boundaries and the actual area calculated by the bounding box is computed. This value can then be used as a confidence measure indicating whether there is only one compact object being tracked which fills the bounding box or multiple small detections. The system also accumulates the x and y positions of each detected pixel. These accumulated sums are then divided by the total number of detected pixels to calculate the centroid of the tracked object. Once it has received an entire frame of data, the system can return the x,y components of the centroid, the four coordinates of color bounding box, the number of detected pixels as well as a confidence value for the object tracked.

Color Statistics. The vision system also includes a color statistic acquisition function. This function keeps a running sum of the individual color channel components. Upon completion of the frame, it divides these accumulated values by the total number of pixels returning the mean color. It also returns an approximation of the absolute deviation from the mean of each color. This can be used like a variance measure to quantify the spread of the colors about the mean. When used in conjunction with other features such as windowing, described below, the color statistics can be used as a building block for a motion detection algorithm or for determining the color of an object at a specific location in the field of view.

Along with the basic algorithmic functions, there are a set of modifying parameters that allow for more advanced image processing. The first of these parameters is the ability to arbitrarily set the window size and location that the user wishes to process. This allows data to be captured in an isolated region of the camera's view. The window bounding box can be easily changed between frames allowing for more localized analysis of the environment. It is also possible to configure the system to return a larger amount of data after each line is processed. This "line mode" can be used in conjunction with both the color tracking algorithm and the statistics function. When used with color tracking, "line mode" will return a binary image of the pixels that fall within the specified color range. Since this data is sent during the delay between the lines of the image, there is no decrease in the overall frame rate. The actual throughput of data does become higher requiring a potentially faster processor to perform any meaningful analysis of the transmitted data. After the binary image is sent, the track color command sends its normal output packet. Figure 3 shows the shape of a hand that had been automatically acquired and then tracked using line mode. When "line mode" is enabled during the acquire statistics function, the mean color value for every image row is sent.

Image Processing and Camera Settings. Another group of functions define how the data is formatted and performs minor adjustments on the overall performance of the system. These functions include a noise filter, an interface transfer flow control setting and a command to modify the CMOS camera's internal image settings. The noise filter mode makes the color tracking algorithm more robust by requiring a valid detection to consist of two horizontally adjacent pixels in the specified color range. This added robustness however can cause small objects not to be detected. The interface flow control settings allow configuration of the serial data entering and leaving the system. The default mode uses visible ASCII characters and continuously streams data as each frame is processed. Selecting "poll mode" instead, causes each function to only return one packet of data and then return to its idle state. This can be useful to help less powerful microcontrollers keep synchronization with the data and can facilitate changing camera parameters between frames. Another setting allows for raw binary bytes to be transferred instead of visible ASCII text and suppresses or enables different synchronization bytes. The camera settings control command allows the user to change the frame rate, toggle white balance, toggle gain, switch between RGB and YUV modes or set any of the OV6620's internal register values.[8]

Demo Mode. To accommodate systems where an extra actuator may be necessary, the camera board has the internal ability to control one standard hobby servo. Using this ability, the vision system can operate in a stand-alone "demo mode". When demo mode is selected, the camera acquires the color of the first object it sees upon power up and tracks it using a simple feedback loop to point a servo toward it. The position of the servo can also be set or read manually, even while demo mode is active. The servo output port can also be used as a TTL digital output instead of to generate a servo PWM signal. For debugging purposes there is also a firmware controlled tracking LED that illuminates when the sensor detects an object. This tracking LED can also be manually controlled via user commands.

2.3 Interface

The vision system by default uses a human readable ASCII communication protocol that allows the user to communicate with it interactively from a serial terminal program. As described previously, a less verbose mode can be enabled to reduce serial port traffic when communicating directly with a computer or another microcontroller. When communicating with a computer, the system can also dump an entire raw image via the serial port. This can be used for diagnostic purposes or higher resolution processing. Due to the high data rate required, a frame dump cannot occur in real time. Instead, the board will send one column of image data per frame that the system processes. At the current default frame rate and maximum window size of 80 × 143 a full frame dump takes about 5 seconds.

By default, all communication with the board takes place at 115.2 kilobaud, but jumpers can be used to select either 9.6, 19.2 or 38.4 kilobaud speeds. Below is an example of a typical set of command transactions with the camera used to track the mean RGB color located in the middle of the image, where the vision system output is shown in italics:

CMUcam v1.12
:**cr 18 44 17 2 19 32**
ACK
:**sw 30 60 50 80**
ACK
:**pm 1**
ACK
:**gm**
ACK
S 150 20 30 5 2 6
:**pm 0**
ACK
:**sw 0 0 80 143**
ACK
:**tc 145 18 24 155 22 36**
ACK
M 50 80 38 82 53 128 35 98
M 52 81 38 82 53 128 35 98
M 51 80 38 84 53 128 35 98

The first command "CR" sets the CMOS camera registers. The numbers that follow are register addresses and parameters, for example 18 44 tells the camera to set the color mode to RGB and turn on automatic white balance. These values are outlined in the vision system documentation [9] as well as the CMOS camera documentation.[8] The "SW" command sets the coordinates of the window to be processed. In this case x1=30 y1=60 x2=50 y2=80, which selects the center of the image. The "PM 1" command turns on the poll mode of the camera so that any additional functions will only return a single line and not stream data. The "GM" command then asks the camera to get the mean value in the current window. The resulting "S" packet shows the R_{mean}, G_{mean}, B_{mean}, followed by the $R_{deviation}$, $G_{deviation}$ and $B_{deviation}$. Next, poll mode is disabled and the window is set back to encompass the entire image. The final "TC" command actually calls the track color function, passing in a minimum RGB value of (145,18,24) and a maximum value of (155,22,36). This value is the mean value previously returned from the camera now padded by its deviation. The returned M packets appear at 16.7 frames per second and show the centroid x, y coordinates the x1, y1, x2, y2 bounding box coordinates, the number of detected pixels and the confidence value of the object being tracked: M x y x1 y1 x2 y2 pixels confidence.

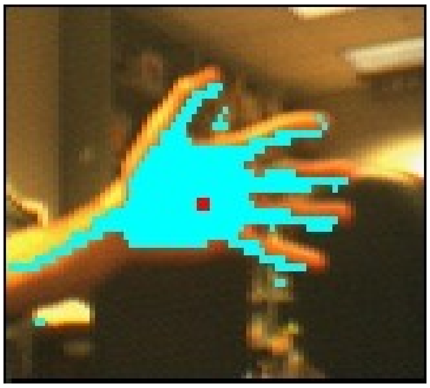

Figure 3: *Example hand image captured by the graphical user interface with tracked bitmap (light turquoise) and centroid (dark red dot) overlaid.*

To aid in system integration, we have also developed a Java based graphical user interface (GUI) that allows the user to interface with the camera from a Unix or Windows based PC. This GUI allows for almost all elements of the camera to be explored in a user friendly environment. The GUI graphically displays real-time data from the camera in a more natural manner. For example, the user can enter color bounds into a dialog box and then issue a track color command. The GUI then formats that request and sends it to the camera. The output of the camera data is parsed and displayed in a window that shows the actual bounding box whose colors depend on the confidence value returned. Depending on how it is configured, the "line mode" binary image and the centroid may also be overlaid on the bounding box, as seen in Figure 3. When the user calls the statistics function, the returned color data is mixed and displayed. One of the most important features of the GUI, is of course, its ability to display a frame dump.

2.4 Performance

Our final vision system operates at a maximum rate of 16.7 frames per second with a maximum resolution of 80×143. Using the Java graphical user interface to display the data, we are able to dump a frame and select the color of an object to track. In one case this object was a blue 14" $\times$ 15" $\times$ 10" recycling bin. Once the object's color bounds are sent to the vision system, it can confidently track the bin up to 35 feet away. From a fixed position, the vision system tracked a stationary 12" x 9" red cardboard target for 10,000 frames. During this period of time the target's center of mass was found to jitter on the x axis and y axis on average by 0.005 pixels and 0.011 pixels, respectively, and with a standard deviation of 0.005 and 0.011 pixels, respectively. The number of pixels tracked was on average 144.00 with a standard deviation of 0.016 pixels. This same test was repeated on a 2" x 2" green target at 10' away. Over the 10,000 frames, the x and y jitter was on average 0.1460 and 0.2900 with standard deviations of 0.146 and 0.216. The average size of the object was 7.98 pixels with a standard deviation of 0.288 pixels. With the appropriate IR coated lens, the system performs well in a wide range of lighting conditions, including direct sunlight outdoors. When the above tests where performed outside during a bright day the results where nearly identical. Video sequences demonstrating various rapid-prototyped mobile robots tracking colored objects using this vision system may be viewed at [10].

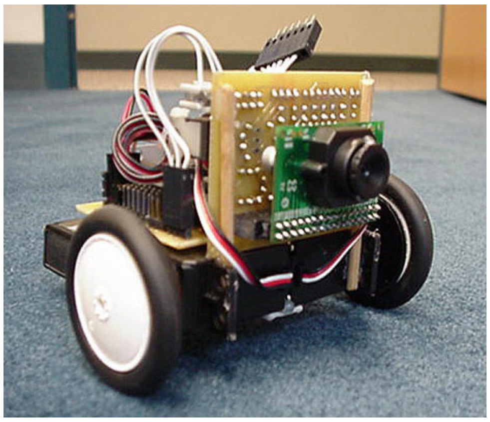

Figure 4: *Picture of the mini-servo robot that uses the vision system for guidance.*

To further evaluate the system we built numerous robotic test platforms. The one shown in Figure 4 consists of two standard hobby servos joined together creating a small differential drive mobile base. A Microchip PIC based microcontroller board is mounted horizontally on top of the servos which then connects to the vision board which is mounted on the front of the robot, perpendicular to the base. A small piece of plastic acts as a sliding caster. A micro-servo connected to this third point of contact allows the robot to tilt up and down. Power is provided by two 9 volt batteries. The main PIC microcontroller communicates with the vision board over a serial link. The PIC first configures the camera to track the brightest red object in front of it. Then, using the coordinates of the object, the robot attempts to center the object in its view. It uses the differential drive base to pan its view left and right. The micro-servo attached to its caster can then push up or down controlling the pitch of the camera. Once the object is centered in the robot's view, it uses the size of the object in order to drive backward or forward in order to hold a constant distance from it. The entire robot measures about 4" by 3" and was less than 3" tall. With all processing and power on board, the robot can successfully track a small brightly colored red doll at distances of up to 15 feet.

3 Related Work

The many hardware and software systems that have been constructed by the computer vision community are too numerous to list here. However, some well known systems have had similar goals to the work described here. The Cognachrome vision system [11] which consists of custom frame grabber and processing hardware has functionality most similar to the system we describe here. The Cognachrome system is definitely more capable than the system described here, it can track 25 objects at 60 Hz. However the system described here is significantly less complex and physically smaller making it more attractive for applications like on board vision for small mobile robots. The MIT Cheap Vision Machine [1] has a similar overall architecture to the Cognachrome system and is similarly more capable than the system described here, but is also significantly more complex. A number of systems [2], [3], [6] consist of highly optimized software systems which rely on standard desktop computer systems to process image data. The system here is unique in that it targets applications where including the capabilities of a standard desktop machine would be prohibitive because of size, cost, or power requirements.

4 Conclusions and Future Work

The goal of this work was to evaluate the feasibility of constructing a minimal vision system consisting solely of a microcontroller and a CMOS camera chip which can implement simple vision algorithms at a useful frame rate. We believe we have demonstrated this feasibility by constructing a functioning system which can successfully find blobs of a specified color in an image at 16.7 frames per second. We further evaluated this system by using it as a sensor to guide several small mobile robots.

There is a great deal of additional functionality that we would like to add to this system, such as the ability to compute color histograms of selected image regions and the ability to perform simple frame differencing. However, this was not possible with our current system due to the limited program and working memory space of the microcontroller used. We have a working prototype of a more powerful system that uses the SX52 microcontroller. This microcontroller is from the same family and has approximately twice the RAM and ROM space than the SX28 model we are currently using. The new prototype system also has a single chip FIFO image buffer that allows for multiple processing passes over a single image. With this new processor and image buffer we hope to significantly enhance the functionality of this system.

Acknowledgments

NASA-Ames provided ongoing funding for this research. Thanks also go to Jon Daley, Shane Keil and Jason Slater who contributed to earlier versions of the vision system.

References

[1] C. Barnhart, The MIT Cheap Vision Machine, http://www.ai.mit.edu/people/ceb/cvm.html

[2] J. Bruce and T. Balch and M. Veloso, "Fast and Inexpensive Color Image Segmentation for Interactive Robots," *Proceedings of IROS 2000*, 2000.

[3] G. D. Hager and K. Toyama, "The XVision system: A general purpose substrate for real-time vision applications," *Computer Vision and Image Understanding*, vol. 69, no. 1, pp. 23-27, January 1998.

[4] I. Horswill, "Polly: A vision-based artificial agent," *The Proceedings of the Eleventh National Conference on Artificial Intelligence*, 1993.

[5] I. Nourbakhsh, D. Andre, C. Tomasi and M. Genesereth, "Mobile Robot Obstacle Avoidance via Depth from Focus," *Robotics and Autonomous Systems*, vol. 22, pp. 151-158, 1997.

[6] K. Konolige, The SRI Small Vision System, http://www.ai.sri.com/˜konolige/svs/

[7] L.M. Lorigo and R.A. Brooks and W.E.L. Grimson, "Visually Guided Obstacle Avoidance in Unstructured Environments," *Proceedings of IROS 97*, pp. 373-379, 1997.

[8] Omnivision Technologies Incorporated, "OV6620 Single-Chip CMOS CIF Color Digital Camera Technical Documentation," http://www.ovt.com/

[9] A. Rowe, C. Rosenberg, I. Nourbakhsh, CMUcam Website, http://www.cs.cmu.edu/˜cmucam/

[10] A. Rowe, C. Rosenberg, I. Nourbakhsh, "A Simple Low Cost Color Vision System," *Technical Sketch Session of CVPR 2001*, 2001.

[11] R. Sargent and A. Wright, "The Cognachrome Color Vision System," http://www.newtonlabs.com/cognachrome/

[12] R. Sargent and B. Bailey and C. Witty and A. Wright, "Dynamic Object Capture Using Fast Vision Tracking," *AI Magazine*, vol. 18, no. 1, 1997.

[13] Ubicom Incorporated, "SX28AC Configurable Communication Controllers Technical Documentation", http://www.ubicom.com/sx/

[14] I. Ulrich and I. Nourbakhsh, "Appearance-Based Obstacle Detection with Monocular Color Vision", *Proceedings of AAAI Conference*, pp. 866-871, 2000.