A Simple Low Cost Color Vision System[*]

Anthony Rowe
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
agr+@andrew.cmu.edu

Charles Rosenberg
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
chuck+@cs.cmu.edu

Illah Nourbakhsh
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
illah+@ri.cmu.edu

Abstract

In this paper we describe a functioning prototype of a simple low cost system which we have constructed and can perform a simple color blob tracking task at 16.7 frames per second. This system utilizes a low cost CMOS color camera module which eliminates the need for a separate frame grabber and all processing is performed by a high speed, low cost microcontroller. The resulting system makes it possible to utilize simple color vision algorithms in applications, like small mobile robotics, where a traditional vision system consisting of a separate camera, frame grabber, and high speed computer would be prohibitive.

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], [6], [8], [9]. 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 run 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 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 chip 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 abilities have increased. This makes it possible to perform simple pixel processing "on the fly" as the pixel values are scanned out of the camera and so a full frame buffer is not necessary. This suggests that it should be possible to team a CMOS camera chip with a low cost microcontroller and implement a simple vision system. We have constructed a functioning prototype system based on this idea which we describe in the remainder of this paper.

[*]This is a revised version of the tech sketch paper accepted to CVPR 2001. A small number of revisions were made in October 2001 to reflect the improved system performance achieved in the month following the original paper submission and correct minor technical errors.

2 System Details

Our vision system is designed to provide high-level information extracted from the camera image to an external processor that may, for example, control a mobile robot. 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. Then, the vision system processes the data in real time and outputs high-level information to the external consumer. The following sections describe the details of the prototype system which we have implemented.

2.1 Hardware System

Our prototype system is a three chip design. The first two chips are the OV6620 CMOS camera and the SX28 microcontroller. The third chip is simply a level shifter for the RS232 serial data. To keep the design simple, the data bus, synchronization pins and configuration bus are directly connected from the OV6620 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 through a Maxim MAX232CPE chip to the outside world via a 115.2 kilobaud serial port implemented in software. We currently have a working stuffed PC board prototype that is 1.75" $\times$ 2.25" and less than 2" deep with the camera module and

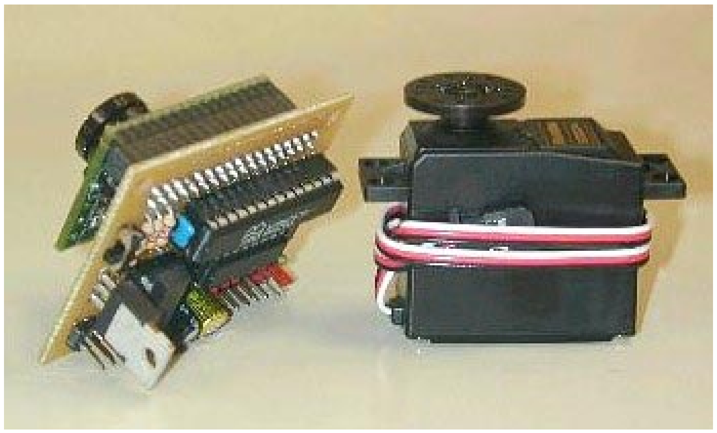

Figure 1: The microcontroller board mated with the CMOS camera module. A standard size hobby servo is shown for scale.

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 relies on an Omnivision OV6620 CMOS camera on a chip. This is mounted on a carrier board which includes the CMOS camera chip, a 4.9 mm F2.8 lens and a few supporting passive components such as a 17 MHz clock crystal. By itself, the board is free running and will output a stream of 8 bit RGB or YUV 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 up to 352×288 with a maximum refresh rate of 60 frames per second. CMOS camera parameters such as color saturation, brightness, contrast, white balance, exposure time, gain and output modes are programmable using a standard serial I2C interface. To utilize video data from the OV6620 one must properly initialize the camera and then must be able to remain synchronized with each of its output signals.

The microcontroller that we use to process the video data is a Ubicom SX28 operating at 75 MHz, model number SX28AC75/DP. It is housed in an easy to solder standard 28 pin narrow DIP package. The SX28 operates at 75 MIPS, 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 "virtual" peripherals.

2.2 Software 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, we had to selectively buffer only part of each image row that is streamed from the camera. We chose to buffer 80 bytes of image data and use the remaining 56 bytes of RAM for processing and other operations. With this mixture we could evenly sample a row of data and had just enough memory left to handle all other operations. Each time the camera sends a row, the 80 byte buffer gets filled with a selection of 20 RGBG pixel packets from the CMOS sensor pixel array. By carefully selecting which bytes of data we buffer and reusing the less important red and blue data, we can effectively get 80 different pixels of horizontal resolution. Since all of the processing is done on a per row basis, the vertical resolution is limited to 143 pixels only by the camera sensor itself. Potentially many different operations could be performed on each 80 pixel group of data. In this first prototype we decided to implement a simple color blob tracking function.

The color blob tracking algorithm allows the user to enter a minimum and maximum bound for each of either the three RGB or YUV channel values, depending on how the camera is configured. Each pixel in the buffer is compared against the user defined 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 only requires that the SX28 store the upper left x1, y1 Cartesian coordinate and the lower right x2, y2 coordinate that enclose pixels which satisfy the color bounds. A running count stores how many pixels actually fell within the color boundaries. Once the entire frame has been processed, some additional post processing operations are done. 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 related to whether there is only one compact object being tracked or multiple small objects. The system finally returns the x1, y1, x2, y2, size and confidence values to the user over the serial port.

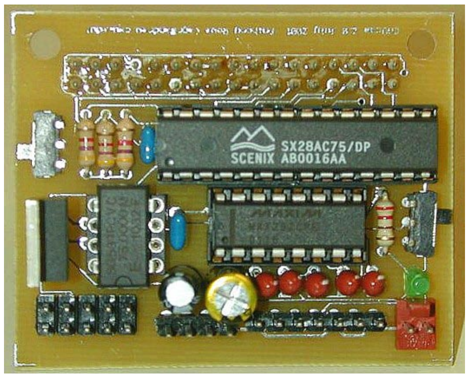

Figure 2: Detail of the assembled microcontroller board, 1.75" $\times$ 2.25". Visible are the microcontroller, the RS232 level shifter, and the clock oscillator.

In this vision system, we also include a mean color function. When used in conjunction with the other features such as windowing, described below, the mean 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.

The vision system uses a human readable ASCII communication protocol that allows the user to communicate to it interactively from a serial terminal program. A less verbose mode can be enabled to reduce serial port traffic when communicating directly to a computer or another microcontroller. When communicating to 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 for such a dump this can not occur in real time. Instead, the board will send two columns of image data per frame that the camera transmits. At the current default frame rate and maximum window size of 80 $\times$ 143 this takes about 5 seconds for a full frame dump. Another useful feature of the system is the ability to do arbitrary windowing in software. This window can be set and changed at any time and applies to all of the image processing functions. The final three built in features include: a noise filter mode, a one shot polling mode, and a modify camera settings command. The noise filter mode makes the color tracking algorithm more robust by requiring two adjacent pixels to be in the specified color range in order to change the bounding box. This added robustness however could cause small objects to be lost. The one shot polling mode, when enabled, causes each function to only return one packet of data instead of streaming packets as the board processes them. This can be used to help less powerful microcontrollers keep synchronization with the data or it could facilitate changing camera parameters between frames. 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 other of the OV6620's internal register values.

By default, all communication with the board is fixed at 115.2 kilobaud, but 38.4 kilobaud can be selected via a jumper setting. The code was all written in C and compiled on ByteCraft's 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.

2.3 Performance

We have not yet performed any extensive quantitative evaluation, but we do have some preliminary ad hoc empirical data. Our final vision system operates at a maximum rate of 16.7 frames per second with a maximum resolution of 80 $\times$ 143. Using a Java graphical user interface to display the data, we where able to dump a frame and pick 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 where sent to the vision system, it confidently tracked the bin up to 35 feet away. We have also successfully tracked objects as small as 2"$\times$ 2" at a distance of 10 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 [7] which consists of custom frame grabber and processing hardware has functionality most similar to the system we describe here. This 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], [5] 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 hopes to target applications where including the capabilities of a standard desktop machine would be prohibitive because of size, cost, or power requirements. In general, although the system described here is not as capable as other systems described in the literature, our goal was to construct as minimal system which was capable enough to be useful in a variety of simple applications.

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 prototype system which can successfully find blobs of a specified color in an image at 16.7 frames per second. We plan to further evaluate this system by using it as a sensor to guide a small mobile robot.

There is a lot of other 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 do simple frame differencing. However, this was not possible with our current system due to the limited program and working memory space of the microcontroller we used. In the future we plan to evaluate a more powerful microcontroller in the same family, the SX52, which has approximately twice the RAM and ROM space and is 33% faster than the SX28 model we are currently using. We chose not to use it in this version because it is only available in a hard to mount surface mount package. With this new processor we hope to enhance the functionality of this system.

Acknowledgements

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

[6] 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.

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

[8] 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.

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