

Using low cost PC based Quickcam video capture device for realtime image processing.

Pongsak Suvanpong (psksvp@ccs.neu.edu)

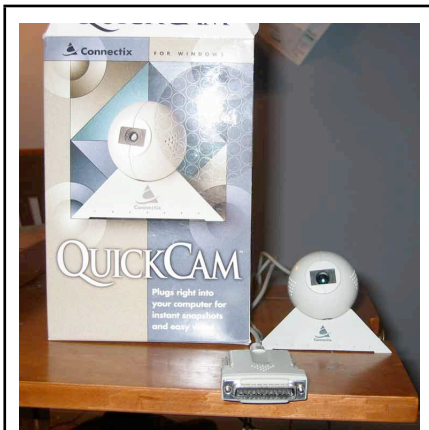
College of Computer Science, Northeastern University, Boston, Massachusetts.

March 13, 1993

Introduction

With the introduction of Quickcam from Connectix Inc, a 486 PC with at least 16 MB of ram, running Windows operating system can capture 8 bits gray scale video at resolution of 320x240 pixels in realtime. The PC version of the Quickcam used parallel port or sometime called printer port to interface with a PC. The power for the device is supplied through PS/2 type keyboard port on a PC. The device has 255 CCD chip from Texas Instruments. It costs less than \$50 from a computer retail shop.

In computer lab at College of Computer Science at Northeastern University, Matrox Inc. Meteor frame buffer was provided for images and video captured. The device was connected with SUN SPARC 10 workstation. This was a very expensive setup. It costed \$3000 just for the Matrox meteor capture device. This was the main motivation for this experiment with a low cost PC based video capture device. We wanted to know if Quickcam was viable to be used to do realtime image processing for robot vision.



Picture 1 shows picture of Quickcam

Methods

To test the Quickcam, a 3x3 Sobel edge detection and blob tracking algorithms were implemented. The algorithms were executed in realtime with each frame captured from a single Quickcam and performance were measured and computed as frame per second(frame rate).

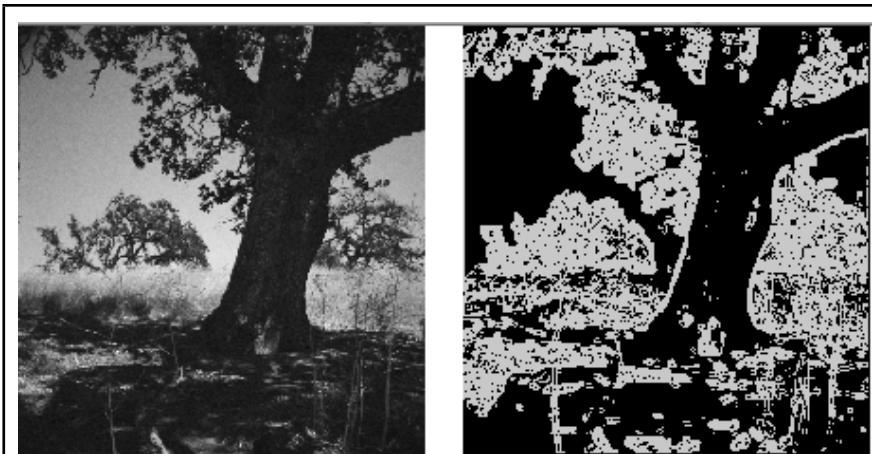
We used a generic laptop PC with 16MB of RAM, 120MB of Hard Disk Drive and a Number Nine Imagine 128 video card. The PC was running Beta version of Windows NT version 4. The compiler that we used to implement the algorithms and test application was Borland C++ compiler version 4.5.

To capture video frame images from Quickcam, Microsoft has created set of APIs called *Video For Windows (VFW)*. The APIs were a set of messages that an application sent to the driver of Quickcam. The process of using these APIs to capture video frame in realtime was roughly described as the following, an application, first, needed to create a video output windows with **capCreateCaptureWindow**. Once that was done, the application could connect with the Quickcam driver using **capDriverConnect** API. Quickcam's driver used callback function to send the video frame image whenever it had one, to set the callback function, we called the API **capSetCallbackOnFrame**. To start the capturing, **capPreview** was called. In the callback function was where we called the test algorithms.

The video frame image was sent using 8 bits lookup table format which each byte in the raw frame image point to a location in a table. Each entry in the table, had RGB value encoded in 32 bits integer. To extract each channel value (red, green, blue), we used the APIs **GetRValue**, **GetGValue** and **GetBValue**. The Quickcam, however, could produce only 8 bits grayscale images, so the value of red, green and blue were all the same in each entry in the lookup table. The range of the value of gray scale intensity was between 0 to 255.

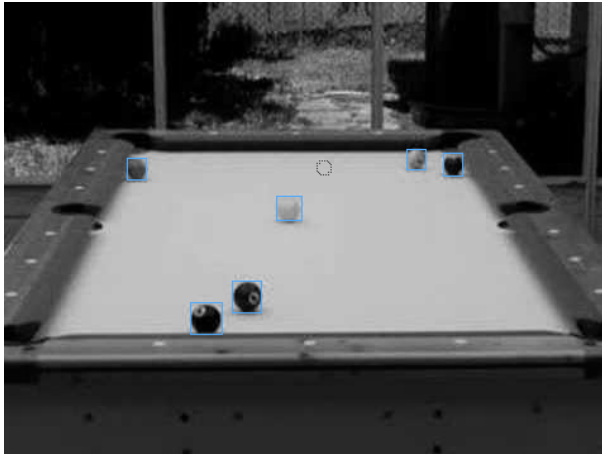
To measure performance, a timer was set to call a function every second. The function, then, recorded the total video frame images that had been processed by the algorithms so far. The Function, then, reset the total back to zero. The performance were measured using 2 videos resolution: 320x240 and 160x120 pixels, captured in realtime.

While measuring performance of Sobel edge detector algorithm, the test laptop and the Quickcam was put on a cart with wheels and pushed around varies places. This was done so that, we could test the algorithm with many different kind of scenes.



Picture 2 shows Sobel algorithm in action, left image shows normal scene and the right image shows scene after running the image through Sobel operator.

For blob tracking algorithm, we pointed the Quickcam at a pool table, and let the algorithm tracked the pool balls while the balls were moving. The Quickcam was mounted about 1 meter at one of the far end of the pool table and angled at about 30 degree looking down, so that the Quickcam could see all of the table surface. We tested the blob tracking algorithm starting from 1 ball to 15 balls. To measure the accuracy of the algorithm, the actual number of balls on the table and number of balls detected by the algorithm were compared in each test sequence. We did total 60 tests, 30 tests at 320x240 resolution and 30 tests at 160x120 resolution.

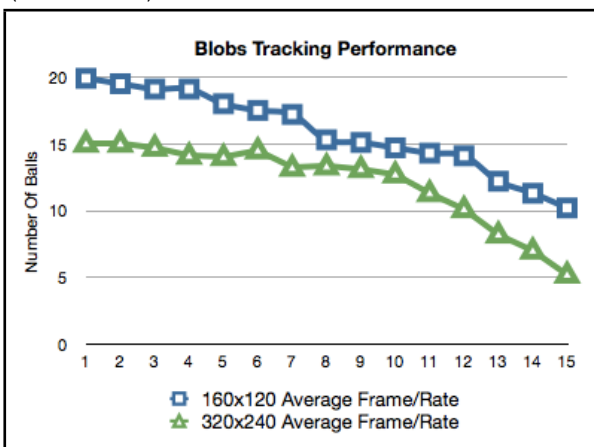


Picture 3 shows blobs tracking algorithm in action. The blue rectangles show the location in an image frame where pool balls were found.

Result and Discussion

The average frame per second for Sobel algorithm was about 16 at 320x240 pixels and 23 for 160x120 pixels. This was the same in all kind of scenes as we expected, since Sobel operator performed 2d spatial gradient which operated on each and every pixels in an image, so scene did not affect the performance of the algorithm. One thing we did not know about the Quickcam was that there was some preprocessing from Quickcam's driver, when we moved the test platform from darker scene to brighter scene, the frame rate was dropped, this was due to the fact that Quickcam driver was doing white balance preprocessing before sending the video frame images to our algorithm. This condition was not presented when we tested the blob tracking algorithm, because the test was done in a constant lighting room.

For blob tracking algorithm, the frame rate of the was varies depending on the number of balls being tracked, the more of the balls on the pool table, the lower of the frame rate. The accuracy of the algorithm were the same with both 320x240 and 160x120 resolution, but at the high resolution(320x240), the frame rate was lower than the lower resolution (160x120).



Picture 4 shows graph of average frame rate per number of balls being tracked.

Conclusion

This experiment had shown that a commodity laptop PC and a low cost video capture device like Quickcam could be used to do realtime image processing. This setup costed the fraction of the cost of the just a Matrox meteor frame buffer and could be used to do research into machine/robot vision algorithm as well. In addition, the new Intel pentium CPU with MMX extension, we could accelerate image processing and machine vision algorithm like the 3x3 Sobel further. The MMX instructions are the data parallel or SIMD(single instruction multiple data) type of instructions. With the SIMD instructions we could pack more than one pixel into a 32 bit MMX register(we could pack 8 pixels in our case with grayscale Quickcam), then an MMX instruction could operate on all of pixels in one time. This kind of processing power was used to be in a very expensive super computer like Thinking Machine CM2.

References

- [1] <http://msdn.microsoft.com>, Programming Video For Windows Documentations.
- [2] Charles Petzold(1992). Programming Windows, 1th Edition
- [3] John G. Scharr(1987). Sobel Algorithm for Edge Detection, Int'l Conf Computer Analysis of Images and Patterns, pp. 207--214, Sep 1990.
- [4] Peter M. Go(1989). Fast Blob tracking using motion estimate, IEEE Trans Image Processing, vol.13(4), pp. 496--508, Apr 1991