SDU International Journal of Technologic Sciences
Vol. 3, No 1/Special Issue, February 2011
pp. 60-68

Computer Technology

COLOR OR MOTION TRACKING USING WEBCAM

Mohammad AZHARİ [*], Cem ERGÜN [**]

Abstract

This study implements a prototype of a motion and color tracking system using webcam. It involves topics on robotics hardware, image processing algorithms and object oriented programming environments such as C# and .NET. The hardware microprocessor was programmed using a C-like language based on CCS Compiler from Microchip Company. All image processing or hardware controls in this project are programmed in a low level manipulation of data which includes pointers. This project only covers a basic implementation of its main goal which is tracking objects. It is highly dependent on the hardware used and calibration and measurements of the system may vary from hardware to hardware.

Keywords: Color Detection, Motion Detection, Image Processing, BLOB Detection, Object Tracking, Servo Controller, PIC Programming, C# Programming

1. Introduction

One of the most discussed sciences in the world is to figure out how human mind works. So far we know so little about human brain functionality when it comes to identification of environment using ears, nose or eyes. A human in a very short moment after birth can start to see the world around him and can identify things and distinguish them. However the way the brain identifies objects is not easy to imitate or simulate using artificial technologies available today, as human vision is based upon top down (TD) scan path theory while image processing is bottom up (BU)[1][2]. One use of image processing for tracking of motion or color is to find an artificial method to simulate gazing or looking of a human eye. However human brain decision of concept to keep focus on is a very complex system [3]. None of the currently available methods are able to guarantee accuracy, considering that there is still debate between top-down and bottom-up theories [1]. We have chosen one of the two basic operations that can be used to at least identify simple objects by their color range or their movement.

The goal of this study is to prototype a real-time system that can pick motion of objects visible to the camera or identify them by their color range and later on, based on the size of the detected objects to rotate the camera using an XY motor system so the detected object would be on focus of our camera. Note that the hardware specification of the system may reflect the level of accuracy of our project. As a result, we are limited to an experimental approach and in future this can be extended to meet practical or commercial expectations.

The rest of the paper is organized as follows. In section 2, we will explain the hardware designed and used for this study as well as its programming. In section 3, we discuss the

[*] Eastern Mediterranean University, Department of Computer Engineering, Famagusta, North Cyprus, E-mail: Mohammad.Azhari@emu.edu.tr

[**] Eastern Mediterranean University, Department of Computer Engineering, Famagusta, North Cyprus , E-mail: Cem.Ergun@emu.edu.tr

software applications that were designed to process the animated images captured by the camera. And finally in section 4 we conclude the study and give a vision of future possibilities.

2. Hardware Architecture and Implementation

2.1. Circuitry

We have used a PIC 18F452 microprocessor and programmed a servo controller. The circuit is responsible for receiving commands and applying the proper signals on the motors. Note that this is a one-way communication. The circuit does not provide any feedback of the result of the motor movements. The functionality schematic of the circuit can be seen in Figure 1.

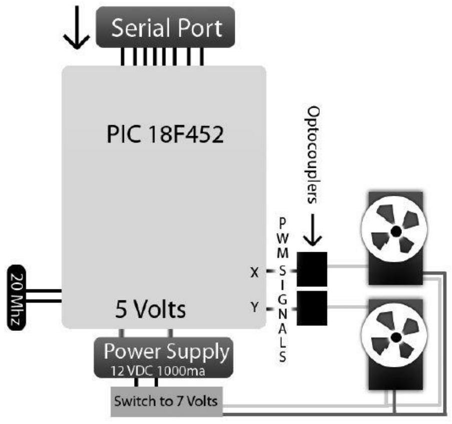


Fig.1. Schematic of the circuit.

2.2. Implementation

There are two things that were considered during the design of the circuit. First of all, boot loader software (provided by Microchip) was injected to the PIC. Afterwards we loaded our software to the chip using the boot loader. This gives us the possibility of changing the chip code using the same serial port we have used for commanding the motors. Thus the external chip programmer is not needed.

The code was compiled using CCS compiler. Another important consideration is that we have limited the input of the circuit so it would not damage the motors by sending out of range data. All data received by serial port is checked before it is sent to the motors.

The technique to control more than one component using a single byte at a time is done by defining external variables in our PIC code. The chip will switch the control between the motors by setting a parameter of “SEND” command. The details of the command can be seen in Table 1.

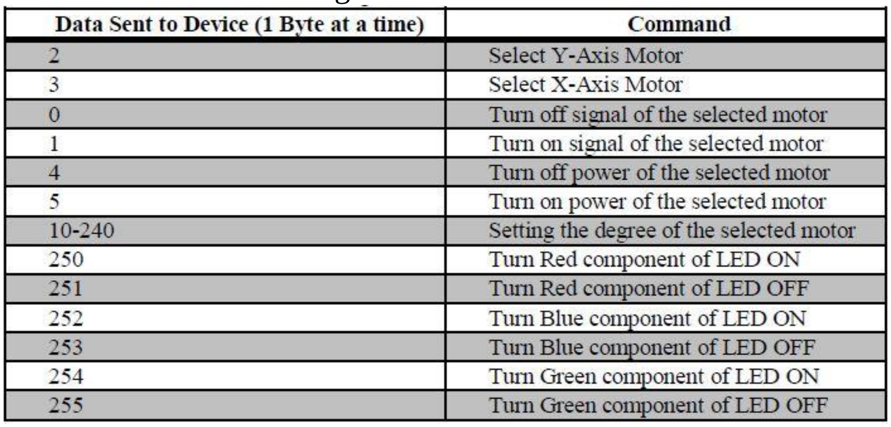
A difficulty we faced during the design of the signal is that using internal PWM signals of the PIC chip was not suitable for our study since laboratory servo motors usually work at 50 Hz frequency. This is too low to be generated by hardware functionality of the PIC chip. Therefore, we chose a software implementation of PWM signals.

An RGB LED is mounted on the bottom of the webcam to be able to identify state of the system at any moment. For example if a moving object or a color is in the sight of the camera the color change**s** to red. We also put a couple of LEDs on the circuit to show the signal passing to the servo motors. Our system is able to receive commands from the serial port for:

- Controlling the degree of both motor on X-axis and Y-axis
- Stopping/Starting the PWM signal on each motor
- Turning on/off the Motor's current
- Controlling the RGB LED

We have created a data sheet for the data received from the Serial port. It only accepts a number between 0 – 255 (See Table 1.).

Table 1. Data Sheet for Commanding the circuit.



Data Sent to Device (1 Byte at a time)	Command
2	Select Y-Axis Motor
3	Select X-Axis Motor
0	Turn off signal of the selected motor
1	Turn on signal of the selected motor
4	Turn off power of the selected motor
5	Turn on power of the selected motor
10-240	Setting the degree of the selected motor
250	Turn Red component of LED ON
251	Turn Red component of LED OFF
252	Turn Blue component of LED ON
253	Turn Blue component of LED OFF
254	Turn Green component of LED ON
255	Turn Green component of LED OFF

3. Software Implementation

The software implementation is based on C# programming language. The implementation is concentrated on applying some of the known techniques of detecting motion or colors without considering the actual shape of objects. This will prevent us from actually following the whole object and will only result in some features being followed as we track those features. Therefore, the motion detection achieved in this study will not detect object completely as in [12] and [13] but only differences sensed by the color change. The color tracking is achieved using only one eye similar to [4] and [6] while some other studies have used two cameras for stereo vision [5].

3.1. Servo Control Module

We have designed a library for controlling the circuit. This module implements ServoSerial Class. The main tasks handled by this module are as follows:

- Getting the list of all ports: gives the list of all ports in the target computer.
- Getting the list of all available ports: gives the list of all ports that currently are not busy.
- Opening or closing a port: used for opening or closing ports.
- Sending a byte to a port: Sends a single byte to the target port.
- Sending commands to motors: Sending commands to motors. This method has arguments such as Motor(X or Y), the target data, delay after sending, auto delay and auto power off.
- Sending commands to change the color of the light under the webcam.

We have also designed a graphic user interface showing the functionality of the ServoSerial library. The interface is responsible for:

- Sending data using a graphical slider
- Defining macros to send different bytes with adjustable delay factor
- Assigning different bytes as shortcut to F keys (F1 to F12)
- Ability to monitor the data sent by using two ports of the computer and a loopback cable.

The first step is to initialize an available port on the computer and then use it to send commands to the circuit. The settings on both the PIC and the software are 9600 Baud rate, 8bit transmission and with no-parity. In case of using USB to serial adapters, the corresponding adapter will automatically create a virtual COM port that can be interacted with just like any other COM port.

The next step is to control the time constraints when sending commands. This means that after sending signals we have to let the motors to reach their destination and this will take some time. Servo Motors will receive a PWM signal and the rest is done by the internal structure of the servo motors which is not controllable by external circuitry.

For the current servos used in this project (Tower Pro 995), the speed rate is 60 degrees per 0.11 seconds when there is no external force on the motor. This however degrades after mounting the camera. Moreover, the force on the X-axis motor causes it to rotate more slowly since the weight of the Y-axis motor is also added.

The method for commanding the motors is designed in a way that is able to calculate the time needed to wait for the motor to finish its job. This method also allows manual control of delays. For example we don’t need for the X motor to finish its job in order to start moving the Y motor. They can both work simultaneously and then a wait loop can be applied after both motors are commanded.

3.2. Joystick Control Module

The Joystick Control Module is designed to work with the ServoSerial Library provided in section 2.2.1. Initially any Playstation 2 compatible can work with this library. In case of other joysticks or game pads we may modify the key codes to work with those controllers.

As all joysticks’ behavior is non-event based, we had to use a timer that ticks periodically and checks for the analogue values of the joystick inputs, and then apply those values to corresponding axis of motors both in speed and direction.

3.3. Image Processing Module

This section explains the algorithms of image processing that are used. For experimentation, initially we have tried very basic algorithms without performance consideration to see results of feature extraction from the images. Later we focused on algorithm improvement. The main operational algorithm of our Color/Motion Tracker can be seen in Figure 2.

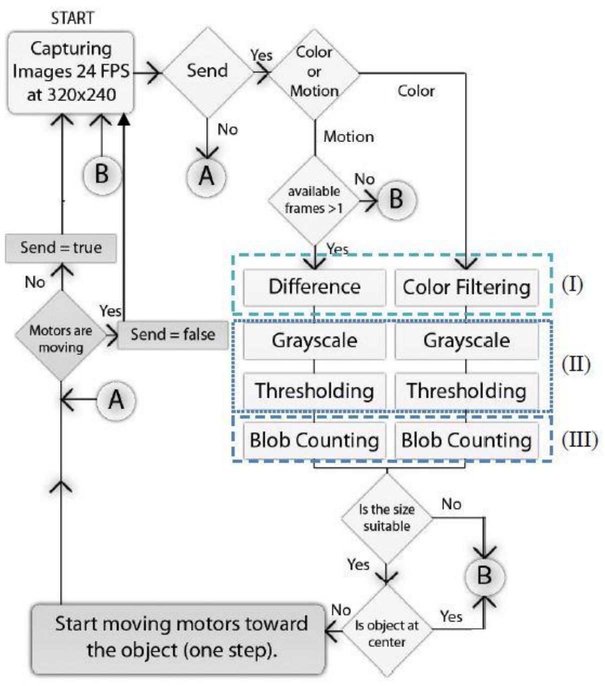


Fig. 2. The main loop of detection and movement: (I) Feature extraction, (II) Reducing calculation redundancies, (III) Object detection.

Initially the image has to be provided by the webcam and after we check if we need it. If not, the webcam will provide us with another picture and we keep checking. In case of needing the image, we decide what filter to go through upon request of the user, and this is where image processing module comes in. We prepare the image for object detection. In case of motion detection we have to have at least three frames to continue. We skip one frame between the two needed ones to increase the detection sensitivity toward slower movements. In case of color detection, we filter the image by a color range.

Two most important steps in both detections are thresholding and blob counting. Thresholding decreases the number of calculations by a factor of 24. Pictures are initially 3 bytes per pixel but after thresholding our pictures will be represented by 1 bit per pixel. This increases the speed of calculations in blob counting step. After the image is processed in either path, Blob Counting will be identical for both. The detected objects by the blob counting method should be checked for size. Some of the objects are detected but are not big enough or are already at the center of image. Therefore we check if our object is at the center, we ignore the detection and process another image provided by webcam. Otherwise, we command the motors for the necessary movements to bring the object at the focus of webcam image. The movement method used in this project is very simple. Note that while motors are moving the images provided by webcam are ignored. Let us take a look at the following algorithm:

```
If the object is at center
        Then ignore it;
Else  { MotorsAreMoving = true;
If object center is at the right of image center
        Then rotate the X motor right;
Else if object center is at the left of image center
        Then rotate the X motor left;
If object center is above image center
        Then rotate the Y motor up;
Else if object center is below image center
        Then rotate the Y motor down;
MotorsAreMoving = false; }
```

Image processing involves complex calculations on large amount of data. Considering the current project as a real-time system, our aim is to satisfy the rule of utilization to have utilization less than 69% to be theoretically safe [7]. The camera sends 24 frames per second of a 320 x 240 size image to be processed by computer for detection. Most of our algorithms use spatial filters. This means that we will visit each image pixel by pixel in order to extract features or enhance the image. This will result in a 24*320*240*3 number of loops per second (3 bytes per RGB pixel). That is 5,529,600 loops per second for each RGB task. As an example, Color tracking will involve 5,529,600 per RGB filter and after gray scaling, 1/24th of that for Blob Counting. This definitely calls for an optimization. We decided to access the bitmap data on a byte level in the memory using pointers. AForge Framework [8], an image processing library for .NET has been exclusively used throughout the experimentation for image processing tasks.

A very important part of our main algorithm for detection is synchronization. All webcams can be interacted through their native drivers. This means we have a limitation on the arrival of images from the webcam. We have to wait for the operating system to provide us with the images. In our case we used the DirectShow webcam wrapper provided as open source by AForge Framework. We do not control the webcam thread that much and we let the driver to send us pictures, and if we are interested we use the image, if not we simply ignore it. The issue is encountered in Motion Detection algorithm. We have to wait for the camera to provide us with at least two sequential frames so we can take the difference.

Another issue is that the motor movements affect the picture of the camera. In this study the camera movement is not tracked and the techniques applied are not relative to the camera tracks. Since a camera's intelligence is a closed world assumption, it does not know when it is moving or when it is not moving. Instead our algorithm would assume the whole world is moving around the camera and not the camera itself. To resolve this issue we had to stop feeding the system with images and wait until the movements of the motors are finalized. After that we can start to send images to the system.

Color Filtering Module.
This module consists of steps to filter the image by the required color ranges. Currently we have only considered one color range. However this can be easily extended to have more than one range.

Note that properties of the webcam may affect the results. Since all webcams come with factory set of Exposures and Low Light Boosting techniques and these properties are changing dynamically, the colors detected may be lost eventually. It is recommended to turn the auto exposure setting off and manually define a constant value for exposure. Low Light Boosting usually decreases the performance of the camera. It is better to turn this one off as well. However, alternatively you may balance and equalize the histograms of the images before feeding them to the filters [9].

If a camera can sense the infrared frequencies, it can be used along with a matrix array of infrared lights to keep the room visible to the camera in case the light in the room is not enough. In most cases a webcam can be disassembled and the IR filter can be taken out so the camera would respond to IR lights. However this may burn the colors toward red.

Motion Detection Module.
This module arranges the image for motion detection. We first wait for the webcam to provide us with at least two frames. Then we take the difference between them. After applying grayscale and threshold filters our object is ready to be passed to Object Detection Module. A similar approach is done for the Color Filtering Module.

Feature Detection Module (Blob Counting).
This module is responsible for gathering information on possible existing objects from the images that are progressed either with Color Detection or Motion Detection modules. Note that we do not consider finding objects as a whole, but to only find the parts that are visible by our desired color filter or the motion in those parts are sensed. It is best to be considered as feature tracking [10] than object tracking. The main method in this module is Blob Counting. The Blob detection will return an array of rectangles and will give information about the position as well as size of features detected. Later on these will be used to move motors accordingly.

3.4. Calibration

One of the difficulties we faced while working with physical devices is that there are error factors involved. From the movements of motors and the time taken to move them to the pictures taken by the camera, we are constantly facing errors that will result in a dysfunctional system. That’s why the calibration of the system is vital. We tried to take measures in different areas in order to make the system usable. One of them is the speed of the motors.

Initially, our circuit cannot understand if the motor has stopped moving or not. So we define a waiting loop after each command of movement. Since we are not touching the voltage value of the motors, the speed is almost constant. In order to move the motors at lower speeds, we move the motors from source to destination step by step (for ex. 1 degree per step) by having a pause loop between the steps. However if the system is measured more accurately we may even be able to consider different speed rates for objects that are at different positions.

However that may require a 3D understanding of the environment which will call for extra AI sensors such as distance detectors. Another applicable approach is to mount two identical cameras to calculate distances using the features after both images taken by the cameras are processed [11].

Currently we have a fixed step for motors. We also consider an error approximation for the objects to be at the center. Our hardware may not be able to perfectly match the center of object with the center of the picture. Instead, we simply ignore the detected feature if it is “almost” at the center. At the moment only the most top-left feature is followed assuming that it is more than the minimum size. If not, it goes to the next most top-left and checks the size again. But this is highly argumentative depending on the type of usage of tracking.

4. Conclusion

In this study robotics-based movement is implemented using servo motors. We also demonstrated movement detection by color and motion features using webcams. Two servo motors along with the PIC chip were used to control the movements. A camera is mounted on those motors and controlled either manually using a USB joystick or automatically by the implemented detection system.

An automatic object follower is implemented that is able to detect the object either by color or motion and to move toward the center of those features accordingly. This project can be extended to support faster or slower moving objects by providing a better hardware. It can also be extended to decisively choose between a group of objects and possibly keeping all of them in sight. Development of motion capturing is also possible by adding some minor steps to the software. This study may further be developed in future to favor objects according to their shapes or textures or even zoom to get a better focus. This project has provided a prototype of a system that can be extended for robotics path finding algorithms as well as distinguishing enemy objects from allies.

Additionally by including a second identical camera in the system, it will be possible to have a stereo vision system which can also sense distance and it will improve the calibration of motor’s movement. We may simply use this as a surveillance system that can be controlled both automatically and manually. It can also be combined with methods of capturing faces for face recognition system or any other image processing based recognition system.

This study may help in designing the Human Machine Interfaces (HMI) for robotics. It can also be combined with real time systems such as computer games to provide a more interesting touch-less interaction in car racing games or educational games for children.

Acknowledgments
The work is supported by Eastern Mediterranean University (EMU) Computer Center and EMU Department of Computer Engineering.

References

[1] Stark, L.W. Privitera, Top-down and bottom-up image processing, C. Neurology & Telerobotics Units, California Univ., Berkeley, CA; Neural Networks,1997., International Conference. Volume: 4, On page(s): 2294-2299 vol.4
[2] Top-down and Bottom-up Design, http://en.wikipedia.org/wiki/Top-down_and_bottom-up_design
[3] Eye Tracking, http://en.wikipedia.org/wiki/Eye_tracking
[4] M. Wasserman, Touchless, http://www.officelabs.com/projects/touchless
[5] A.Kirillov, Making a step to stereo vision, http://www.aforgenet.com/articles/step_to_stereo_vision
[6] A.Kirillov, Lego Pan Tilt Camera and Objects Tracking, http://www.aforgenet.com/articles/lego_pan_tilt_camera
[7] Phillip A. Lapante., Real-Time System Design and Analysis 3rd Ed.
[8] AForge Framework, http://www.aforgenet.com

[9] Uemura, H.; Joo Kooi Tan; Ishikawa, S.; A Color Tracker Employing a Two-dimensional Color Histogram Under Changeable Illumination, IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on 6-10 Nov. 2006 Page(s):3273 – 3278.

[10] D. Chetverikov and J. Verest´oy, Feature Point Tracking for Incomplete Trajectories, Budapest, Computing 62, 321–338 (1999).

[11] Rafael Munoz-Salinas, Eugenio Aguirre, Miguel Garci´a-Silvente, People detection and tracking using stereo vision and color, Image and Vision Computing 25 (2007) 995–1007.

[12] Wu Huimin; Zheng Xiaoshi; Zhao Yanling; Li Na; A New Thresholding Method Applied to Motion Detection, Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on Volume 1, 19-20 Dec. 2008 Page(s):119 – 122.

[13] Meng-Chou Chang; Yong-Jie Cheng; Motion Detection by Using Entropy Image and Adaptive State-Labeling Technique, Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on 27-30 May 2007 Page(s):3667 – 3670.