RGB Color Detection Using MATLAB from Live Video

[Control Theory (EI502)]

POWERED BY Math Works

Submitted by

Tulika Biswas(11705516004)

Suvam Saha(11705516007)

Subham Hati(11705516012)

Sourabh Kumar Bania(11705516017)

Rajarshi Roy (11705516031)

Faculty in charge: Ms. Naiwrita Dey

Assistant Professor

Dept. of AEIE, RCCIIT

RCC Institute of Information Technology

Beliaghata, Canal South Road, Kolkata-700051 [Academic Session: 2018-2019]

CONTENTS

ABSTRACT

RBG Color detection algorithm in this project is used to detect red, green and blue color objects from an image frame captured using webcam. The objects found and enclosed in rectangles. The algorithm also calculates the number of red, green or blue objects and if one or more such objects are found then red, yellow or green LED glows respectively with help of arduino. The code captures 100 frames from the video feed and performs the color detection and object counting scheme on each frame. The object color and centroid position of the object is displayed on the MATLAB interface for each frame.

INTRODUCTION

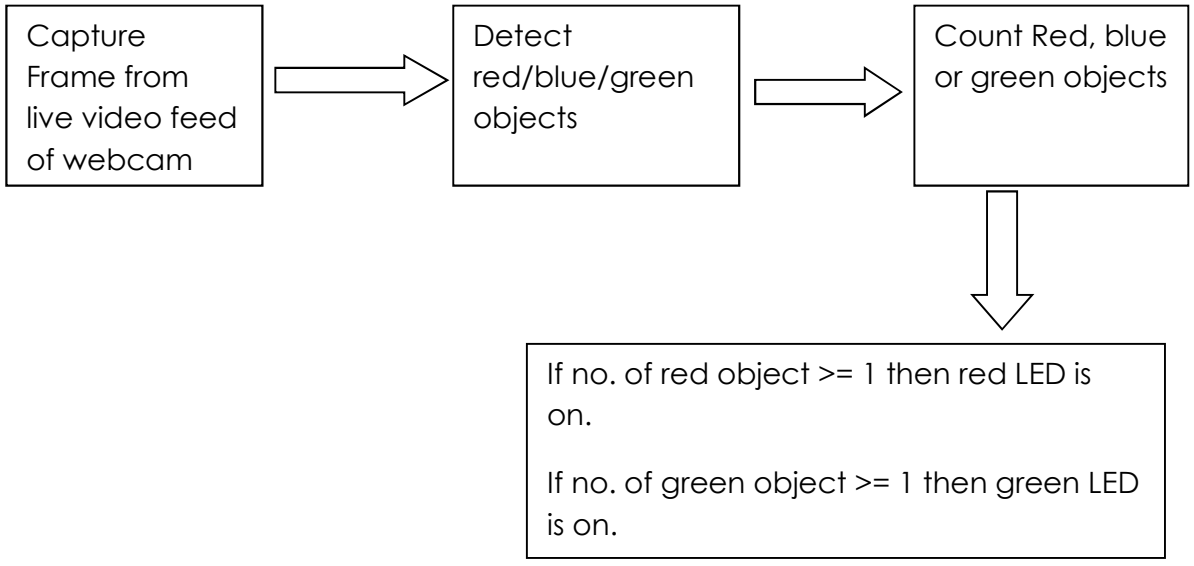
This project uses the webcam to capture a video feed and the MATLAB interface takes each frame and uses the algorithm to detect Red, Green and Blue colored objects in each frame. For each object, the centroid is detected, and each detected object is enclosed in the best fit rectangle of the respective color. If number of red, blue or green object is at least one then red, yellow and green LED glows respectively.

METHODOLOGY

Algorithm:

1. Capture a frame from live video
2. Detect red/blue/green colored object using Kernel and subtract grayscale frame from it
3. Remove noise using Median filter
4. Convert the filtered image to binary
5. Complement the binary image and count objects
6. Send the data to the arduino to glow the LED
7. Remove pixels less than 300px from the binary image
8. Label each object
9. Display the Centroid location and bounding rectangle of each object
10. Repeat step 1-9 for 100 frames

Block Diagram:



ARDUINO TOOLBOX:

MATLAB® Support Package for Arduino® Hardware enables you to use MATLAB to communicate with an Arduino board. You can read and write sensor data through the Arduino and immediately see the results in MATLAB without having to compile. This support package is functional for R2014a and beyond.

Simulink® Support Package for Arduino® Hardware enables you to create and run Simulink models on Arduino boards. The support package includes a library of Simulink blocks for configuring and accessing Arduino sensors, actuators, and communication interfaces. It also enables, you to interactively monitor and tune algorithms developed in Simulink as they run on Arduino.

Capabilities and Features:

With MATLAB® Support Package for Arduino®Hardware, you can use MATLAB to interactively communicate with an Arduino board. The package enables you to perform tasks such as:

Acquire analog and digital sensor data from your Arduino board

Control other devices with digital and PWM outputs

Drive DC, servo, and stepper motors (also supports Adafruit Motor Shield)

Access peripheral devices and sensors connected over I2C or SPI

Communicate with an Arduino board over a USB cable or wirelessly over Wi-Fi. Build custom add-ons to interface with additional hardware and software libraries. Because MATLAB is a high-level interpreted language, you can see results from I/O instructions immediately, without compiling. MATLAB includes thousands of built-in math, engineering, and plotting functions that you can use to quickly analyze and visualize data collected from your Arduino.

IMAGE PROCESSING: is a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. You can perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing.

Image Processing Toolbox apps let you automate common image processing workflows. You can interactively segment image data, compare image registration techniques, and batch-process large data sets. Visualization functions and apps let you explore images, 3D volumes, and videos; adjust contrast; create histograms; and manipulate regions of interest (ROIs).

You can accelerate your algorithms by running them on multicore processors and GPUs. Many toolbox functions support C/C++ code generation for desktop prototyping and embedded vision system deployment.

Hardware Prototype

The Hardware Used:

1.Arduino

2.Breadboard

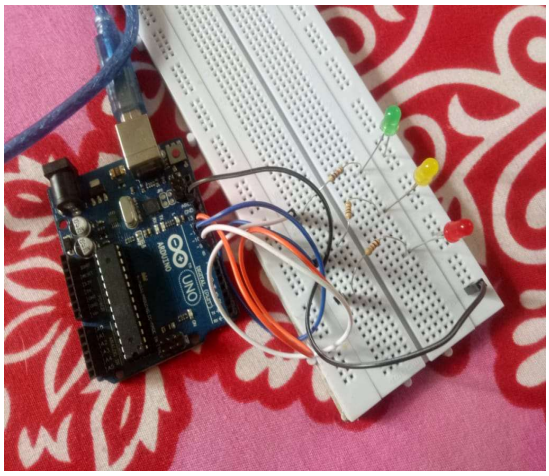
3. LED

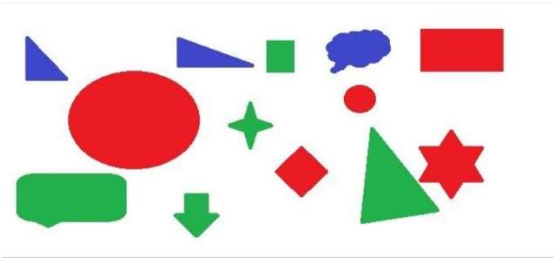
4.3pc Connectors

5.HP True Vision Camera (2 megapixel)

EXPERIMENTAL RESULTS

The following images shows how RGB colours are detected on the screen through Arduino using MATLAB interface and results are obtained.





CONCLUSION

The successful results conclude that we have been able to detect RGB colours and provide the necessary output through LED signals as it detects colour.

The MATLAB Interface and Image processing toolbox has helped to guide our experiment and also special thanks to our subject teacher Ms. Naiwrita Dey (Assistant Proffessor AEIE Dept) for letting us to work on this project and guiding us through doubts and difficulties we faced .

FUTURE SCOPE

The future of image processing will involve scanning the heavens for other intelligent life out in space. Also new intelligent, digital species created entirely by research scientists in various nations of the world will include advances in image processing applications. Due to advances in image processing and related technologies there will be millions and millions of robots in the world in a few decades time, transforming the way the world is managed. Advances in image processing and artificial intelligence6 will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport. With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance, and the visual system of man can be replicated. The future trend in remote sensing will be towards improved sensors that record the same scene in many spectral channels. Graphics data is becoming increasingly important in image processing app1ications. The future image processing applications of satellite-based imaging ranges from planetary exploration to surveillance applications.

REFERENCES

MATHWORKS MATLAB WEBSITE.

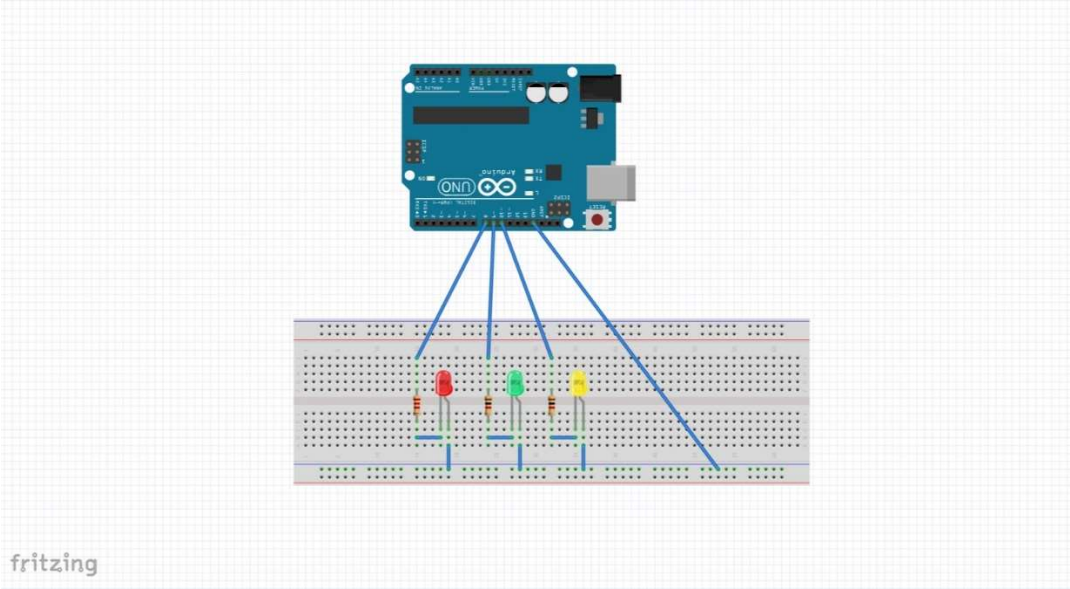
2.YOUTUBe VEDIOS.

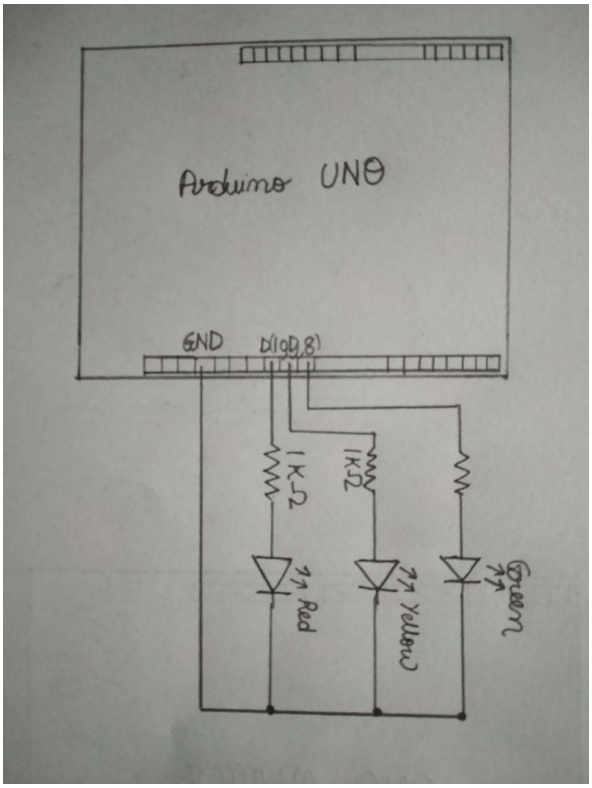
3.ARDUINO OFFICIAL WEBSITE.

4.All About circuits (WEBSITE).

5.Fritzing.

Appendix A: Circuit Diagram





APPENDIX B: CODE

```
obj=videoinput('winvideo',1,'yuy2_640x360');
obj.ReturnedColorspace = 'rgb';
B=getsnapshot(obj);

framesAcquired = 0;
x=arduino();
while (framesAcquired <= 100)
   data = getsnapshot(obj);
      framesAcquired = framesAcquired + 1;
     diff_imR = imsubtract(data(:,:,1), rgb2gray(data));
      diff_imR = medfilt2(diff_imR, [3 3]);
      diff_imR = im2bw(diff_imR,0.18);
      %stats = regionprops(diff_im, 'BoundingBox', 'Centroid');

      %counting red objects
      diff_imR2=~diff_imR; %complement
      CountR = bwboundaries(diff_imR2);
      if length(CountR) > 1
         writeDigitalPin(x, 'D10', 1);
      else
         writeDigitalPin(x, 'D10', 0);
      end
   diff_imG = imsubtract(data(:,:,2), rgb2gray(data));
      diff_imG = medfilt2(diff_imG, [3 3]);
      diff_imG = im2bw(diff_imG,0.18);
      %counting Green objects
      diff_imG2=~diff_imG; %complement
      CountG = bwboundaries(diff_imG2);
      if length(CountG) > 1
         writeDigitalPin(x, 'D8', 1);
      else
         writeDigitalPin(x, 'D8', 0);
      end
     diff_imB = imsubtract(data(:,:,3), rgb2gray(data));
      diff_imB = medfilt2(diff_imB, [3 3]);
      diff_imB = im2bw(diff_imB,0.18);

     %counting blue objects
      diff_imB2=~diff_imB; %complement
      CountB = bwboundaries(diff_imB2);
      if length(CountB) > 1
         writeDigitalPin(x, 'D9', 1);
      else
         writeDigitalPin(x, 'D9', 0);
      end
     % Remove all those pixels less than 300px
      diff_imR = bwareaopen(diff_imR,300);
      diff_imG = bwareaopen(diff_imG,300);
      diff_imB = bwareaopen(diff_imB,300);
    % Label all the connected components in the image.
```

```
     bwR = bwlabel(diff_imR, 8);
     bwB = bwlabel(diff_imB, 8);
     bwG = bwlabel(diff_imG, 8);
     % Here we do the image blob analysis.
    % We get a set of properties for each labeled region.
    statsR = regionprops(bwR, 'BoundingBox', 'Centroid');
    statsG = regionprops(bwG, 'BoundingBox', 'Centroid');
    statsB = regionprops(bwB, 'BoundingBox', 'Centroid');
    % Display the image
    imshow(data)
    hold on
    %This is a loop to bound the red objects in a rectangular box.
    for objectR = 1:length(statsR)
        bbR = statsR(objectR).BoundingBox;
        bcR = statsR(objectR).Centroid;
        rectangle('Position',bbR,'EdgeColor','r','LineWidth',2)
        plot(bcR(1),bcR(2), '-m+')
        a=text(bcR(1)+15,bcR(2), strcat('X: ', num2str(round(bcR(1))), '
Y: ', num2str(round(bcR(2))), '    Color: Red'));
        set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,
'Color', 'red');
     end
   for objectG = 1:length(statsG)
        bbG = statsG(objectG).BoundingBox;
        bcG = statsG(objectG).Centroid;
        rectangle('Position',bbG,'EdgeColor','g','LineWidth',2)
        plot(bcG(1),bcG(2), '-m+')
        b=text(bcG(1)+15,bcG(2), strcat('X: ', num2str(round(bcG(1))), '
Y: ', num2str(round(bcG(2))), '    Color: Green'));
        set(b, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,
'Color', 'green');
    end
   for objectB = 1:length(statsB)
        bbB = statsB(objectB).BoundingBox;
        bcB = statsB(objectB).Centroid;
        rectangle('Position',bbB,'EdgeColor','b','LineWidth',2)
        plot(bcB(1),bcB(2), '-m+')
        c=text(bcB(1)+15,bcB(2), strcat('X: ', num2str(round(bcB(1))), '
Y: ', num2str(round(bcB(2))), '    Color: Blue'));
        set(c, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,
'Color', 'blue');
         end
    hold off
    end
clear all
```

Appendix C: Data sheet of sensors

Video Resolution	720p (1280*720) cmos sensor
Focus Method	Fixed Focus
usb certification	usb 2.0 high speed certified
Video resolution support	Live video resolution supported up to 30 fps
Still Picture Resolution	Up to 5.7 megapixel