AINT308 - OpenCV Assignment 1 2022

Student No. 10618407
School of Engineering,
Computing and Mathematics
University of Plymouth
Plymouth, Devon

***Abstract*—Machine learning is not a new technology, it is a field that is becoming more prevalent within modern engineering practises. It is being used more in the rapidly evolving fields of autonomy and automation. This report outlines some of the functionality of a popular C based computer visions library OpenCV. Task 1 is to evaluate the colors of pixels in a picture to determine the colour of a given object in the frame (car). Task 2 was to track on object across frames of a video to track its motion (swinging pendulum). Task 3 was to identify and cross correlate components on a circuit to check for any missing components.**

Keywords:

Computer Vision, OpenCV, Object Detection, RGB

I. Task 1: Colour Sorter

A. Introduction

The first task was to evaluate the colour of a given object in a given frame color within the RGB colour space. The task was to identify the colour of a given car. An example of the type of image can be seen in Fig 1.

Fig. 1: Example Car Image

B. Solution

Fig 2 below shows my solution to the first task, the Colour Sorter.

```
for (int i = 0;i<Car.rows; i++) { // runs through all the rows in the image
    for (int j = 0;j < Car.cols; j++) { // runs through all the columns in the image
        Vec3b PixelValue = Car.at<Vec3b>(i,j); // stores the RBG values in the PixelValue vector

        // checks if the blue in the pixel is at least 1.5x higher than the other colours
        if((PixelValue[0] > 1.5 * PixelValue[1])&&(PixelValue[0] > 1.5 * PixelValue[2])){
            // pixel is blue
            blueCount++; // add the pixel to the blue count
        }
        // checks if the green in the pixel is at least 1.5x higher than the other colours
        else if((PixelValue[1] > 1.5 * PixelValue[0])&&(PixelValue[1] > 1.5 * PixelValue[2])){
            // pixel is green
            greenCount++; // add the pixel to the green count
        }

        // checks if the red in the pixel is at least 1.5x higher than the other colours
        else if((PixelValue[2] > 1.5 * PixelValue[1])&&(PixelValue[2] > 1.5 * PixelValue[0])){
            // pixel is red
            redCount++; // add the pixel to the red count
        }
        else {
            // inconclusive - do not add any values to the count
        }
    }
}
```

Fig. 2: Task 1 Code - Colour Checking

The code iterates through two loops, one to look at all the rows and the second to go through all the columns in a given row. The value of each pixel is the evaluated, if the R(red), G(green), or B(blue) value if more than 1.5 times larger than the other R, G, B values, then the pixel is deemed to be that colour. The colour of this pixel is added to a running total for each of the colours.

After the whole image is checked the values are checked and the highest value is deemed to be the colour of the car. This can be seen in Fig 3.

```
    cout<<"The blue value is " << blueCount <<endl;
    cout<<"The green value is " << greenCount <<endl;
    cout<<"The red value is " << redCount <<endl;

    // if blue is the higest count the car is blue
    if((blueCount > redCount)&&(blueCount > greenCount)){
         cout<< "The car is blue" <<endl;
    }

    // if green is the higest count the car is green
    else if((greenCount > redCount)&&(greenCount > blueCount)){
        cout<< "The car is green" <<endl;
    }

    // if red is the higest count the car is red
    else if((redCount > blueCount)&&(redCount > greenCount)){
        cout<< "The car is red" <<endl;
    }

    //display the car image untill x is pressed
    while(waitKey(10)!='x'){
        imshow("Car", Car);
    }

}
```

Fig. 3: Task 1 Code - Output

Overall this method worked for the dataset that was provided. This dataset was selected so that the car made up a majority of the frame and the background were, for the most part plain and didn't sway the results.

C. Further Improvements

Although this task worked for the given dataset, there are some ways to improve the accuracy. RGB models are typically not used for this type of task, with the Hue, Saturation, and Value (HSV) model being preferred. Initially the RGB model was built and optimised to be used on display screens, whereas the HSV model was developed to mimic how a human interprets colours. Since HSV models are able to separate colour vales and lightness, operations can be more easily performed on the hue itself. Because we are more interested in the colour of the cars and not the lightness, a HSV colour model would be a objectively better solution. [1]

Another improvement is to make the algorithm detect the outline of the car. This would eliminate the issues caused by having a background that could be mistaken for the body of the car. An example of this can be seen in Fig 4.

Fig. 4: Example Car with undesirable background

The red car with a green background may confuse the simple sorting algorithm and lead to a false result (the algorithm may think the car is green). This is the issue with using such a basic classification algorithm.

The final improvement that could be partially solved by using the HSV colour model, but that would also require a far more complex solving algorithm, is that cars are not just red, green, or blue. They come in many different colours (including white, black, and silver) and also in many different shades and brightness of colour. By isolating just the RGB values and evaluating which is highest, a lot of granularity is lost. Using this method it is impossible to detect the shade of the car and even the type of red, green, or blue the car may be, if it is one of these colours at all. Using the blend of colours and evaluating them against known RGB values, further colours of car could have been identified with a significantly higher range of colours being available (Up to 16M different combinations for an 8-bit RGB value). This method however would require some form of 'lookup table' that would be used to compare the RGB pixel value to. This adds complexity to the task, but allowed for a better overall result.

D. Conclusion

Overall this task was successful for the limited dataset that was provided. This dataset only included Red, Green, and Blue cars. These cars were placed in centre of the frame with neutral lighting. Due to the limited natured of both the colour space sued and the approach, anything more substantial than minor performance increased would require a overhaul of the methodology used. Changing from the *RGB* to the *HSV* colour space would increase the accuracy of the colour detection and would allow the solution to be expanded to a fuller range of car colours. Finally, outlining the car and omitting the background would reduce the likelihood of the background of the image influencing the detected colour of the car.

II. Task 2: Colour Tracker

A. Introduction

B. Solution

The figure below shows my solution to the second task, the Colour Tracker.

C. Further Improvements

D. Conclusion

III. Task 3: Cross Correlation

A. Introduction

B. Solution

The figure below shows my solution to the third task, the Cross Correlation.

C. Further Improvements

D. Conclusion

References

[1] T. Mamdouh. (2020, Apr.) Color spaces (rgb vs hsv) - which one you should use? [Online]. Available: https://discover.hubpages.com/technology/Color-spaces-RGB-vs-HSV-Which-one-to-use

The code can be found on GitHub Here!