Aint308 Assignment 1 OpenCV

By Zil-E-Hasnain Shah

Abstract

A OpenCV Solution has been developed to aid the client's needs for colour detection, object tracking and image matching. Further in this report these methods will be elaborated and explained in detail, to ensure clarity of colour detection, object tracking and image matching. This will be supported through evidence, within academic research, flowcharts, codes and analysis.

1.1 Task 1 – Colour Sorter

An online car dealership has requested a computer vision solution to identify the colour of each car in their database. In this task a solution is designed that identifies the colour of each car via Red Green and Blue (RGB) pixels and the outcomes of the results to the terminal. Rapid tables offer a visual colour model to give the audience to visual aid see the colour based on the RGB code (No date: Unpaginated) [1].

1.2 Task 2 – Colour Tracker

A researcher is using a pendulum for an experiment and wants a design to create a solution that can observe the decay rate of the oscillation and graph out the results in small time intervals. This task aims to identify the moving colour target frame by frame and create a program that can measure the angle of the pendulum over time into a data file. This can be used at a later stage to find out the friction of the pivot point in the pendulum.

1.3 Task 3 – Cross Correlation

A Printed Circuit Board (PCB) manufacturer wants to automate part of their quality control, as a few key components have a high failure rate during the pick and place phase in production. A solution will be developed that can output the error of the position and detect any quality degradation via template matching.

2. Design

2.1 Task 1 – Colour Sorter

Using a computer vision solution, the task request is to identify the colour of each car in the database. An initial brainstorm flowchart can be seen in Figure A. Figure A explains how the program should essentially work. First of all, there is a fixed set of images of the vehicles and a specific image size of 640x480 pixels. Using the built-in functions of OpenCV platform, we can use a function to calculate each independent RGB profile value of a single pixel. Subsequently, cycling this through a loop we can process each individual pixel, to take this further and introduce a counter that can keep track of each RGB individual colour value. With this, the measurements can be accumulated over in order to find the dominant RGB pixel value. This is because each pixel has a range of Red, Green and Blue values which makes up to 16777216 different combinations [1].

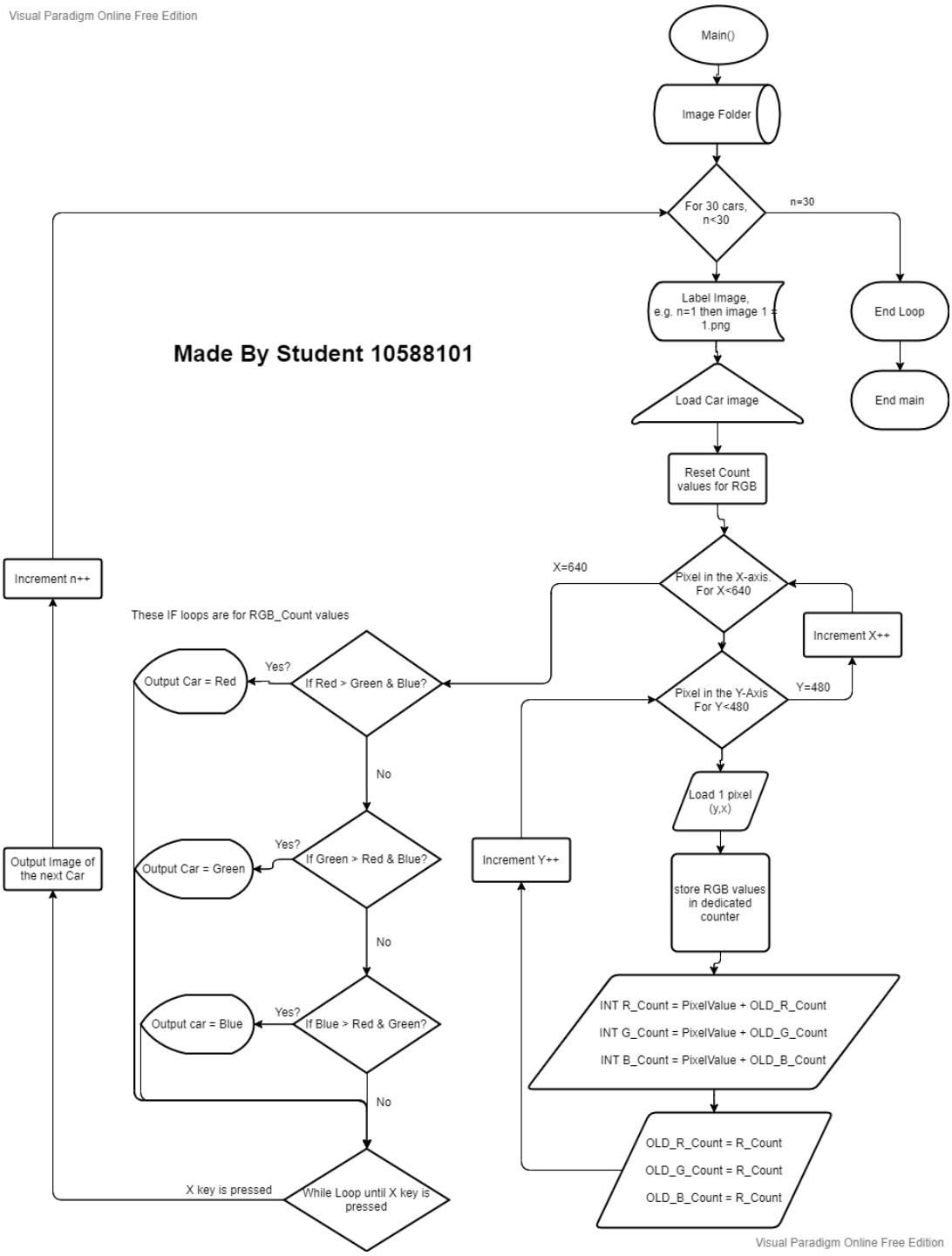

Figure A - Flowchart showing how the code operates. [A]

2.2 Task 2 – Colour Tracker

Provided with video data from the researcher, an observation of the pendulum movement can help calculate the decay rate of the oscillation measured at small time intervals to calculate the angles using trigonometry. One such way of doing this is by calculating Moments between the pivot and the pendulum object as it is oscillating.

Since the oscillating object is bright green in colour, a computer program can detect and calculate without much needed human intervention [2]. An alternative method of detection is by cutting the video into frames, these are still images which would make it easier to work with. Once this storage class contains these frames we can start converting the colour subspace from RGB (or BGR) to Hue, Saturation, Value (HSV) format by using the cvtcolor function and colour conversion BGR2HSV In reference to an online documentation, colour space conversion is when RGB are converted to a floating point format from 8 or 16 bit images and then scaled to fit the 0 to 1 range, the mathematical formula is shown on the reference page...(Colour Space Conversion: No date: Unpaginated) [5]. This gives the image a hue range of H between 0 and 180 for an 8-bit image. The next step is to filter out the HSV frames to grayscale by using the inrange function and selecting lower and upper bound range of colour. In this case since the oscillating object is green, lower bound would be near black whilst upper bound would be close to green. This will allow the program to filter out everything but green. Further brainstorming leads to flowchart for the program solution as shown in Figure B. In Anjana Pillais report *Using Atan2 to find Angles in OpenCV, is there Another Method? The* author explains how to measure the angle of the pendulum using the atan2 method with responses to her enquiry from fellow peers (2017: Unpaginated) [4] as guideline the method is quite clear and concise with ease of use.

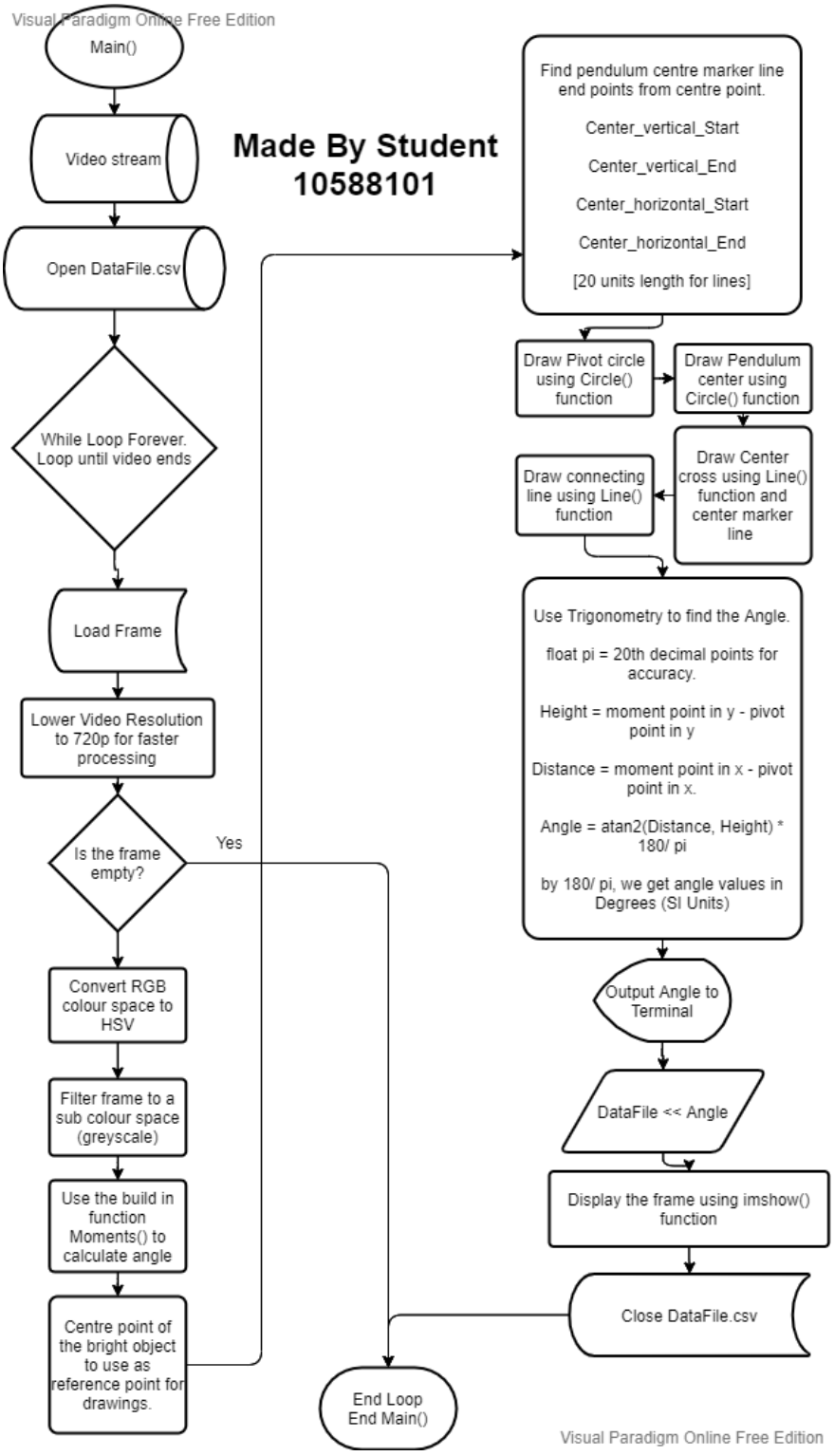

Figure B - Flowchart showing how the code operates [B]

uses built in function moment() following the basics guideline by James Roger [7] to Calculate all of the moments up to the third order of a polygon or rasterized shape and returns an array of 2D points.

```
//Trignomentry to find angle
float pi = 3.14159265358979323846; // Value of pi
double Height = p.y - Pivot.y; // length of pendulum rod (red long line)
double Distance = p.x - Pivot.x; // horizontal arc distance (not displacement)
double Angle = atan2(Distance, Height) *(180 / pi); // use atan2 with arc & height to get angles in rads-1 and then convert to deg.
```

Figure 1 - Snippet of the code.

In this code, it's shown that the angle is calculated using the atan2() function in terms of trigonometry.

2.3 Task 3 – Cross Correlation

Task 3 works by template matching, this is the process of matching the input image with the template image, this can be seen in the online source, OpenCV where they explain how this method works (Template Matching: No date: Unpaginated) [2].

There are 6 known functions of template matching. 1 key example is in the format of TM_SQDIFF_NORMED (Object Detection: No date: Unpaginated) [3], which is used to find the x and y coordinates of the template image.

Using object cross correlation, an image or video can accurately be overlapped and be pinpointed to a selected object and their location. This line of code "matchTemplate(PCB, Component, Mask, TM_SQDIFF_NORMED);" can be explained by using a source image and a secondary image to overlap each other, it tries to match

the secondary image to the parts in the source image. using the SQDIFF_NORMAD method to get values of the closest match possible and create a Mask Layer.

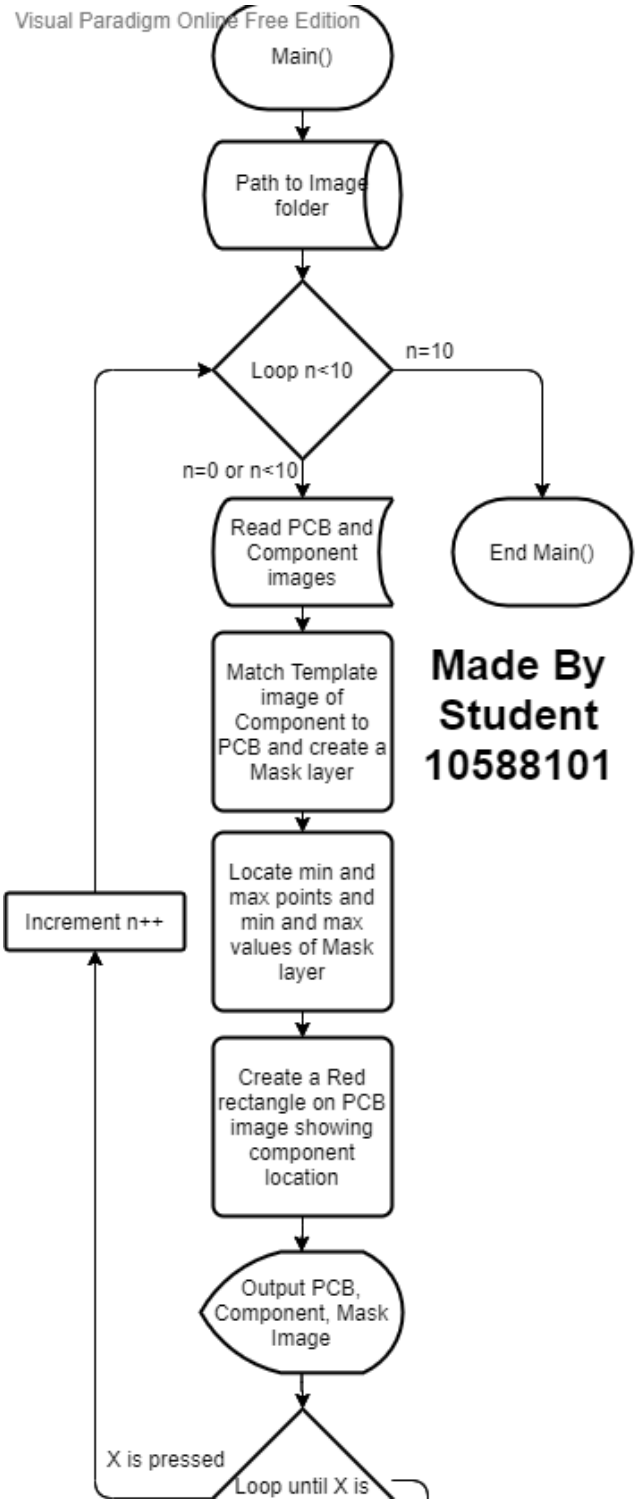

Figure C- Flowchart showing how the code operates [C]

"minMaxLoc(Mask, &*Min_Val*, &*Max_Val*, &*Min_Loc*, &*Max_Loc*);"

This line of code returns the location of the min and max location values that are needed to draw a red rectangle. which then is shown on the output image.

An initial plan was drawn up as a flowchart which is shown in Figure C.

3. Results

3.1 Task 1 – Colour Sorter

A video [D] of the results has been provided.

Stack Overflows offers a solution to detect primary RGB colours. Using their programming software C++ on OpenCV, they show how to separate colours based on their RGB values. This can be used to divide between their upper and higher values in order to precisely identify the colour of the car (Detect RGB Colour Interval: No date: Unpaginated) [6].

3.2 Task 2 – Colour Tracker

The solution as shown in the Video [E] with the result shown in the graph below Figure

2, Over time the pendulum reaches equilibrium which is shown in Figure 2.

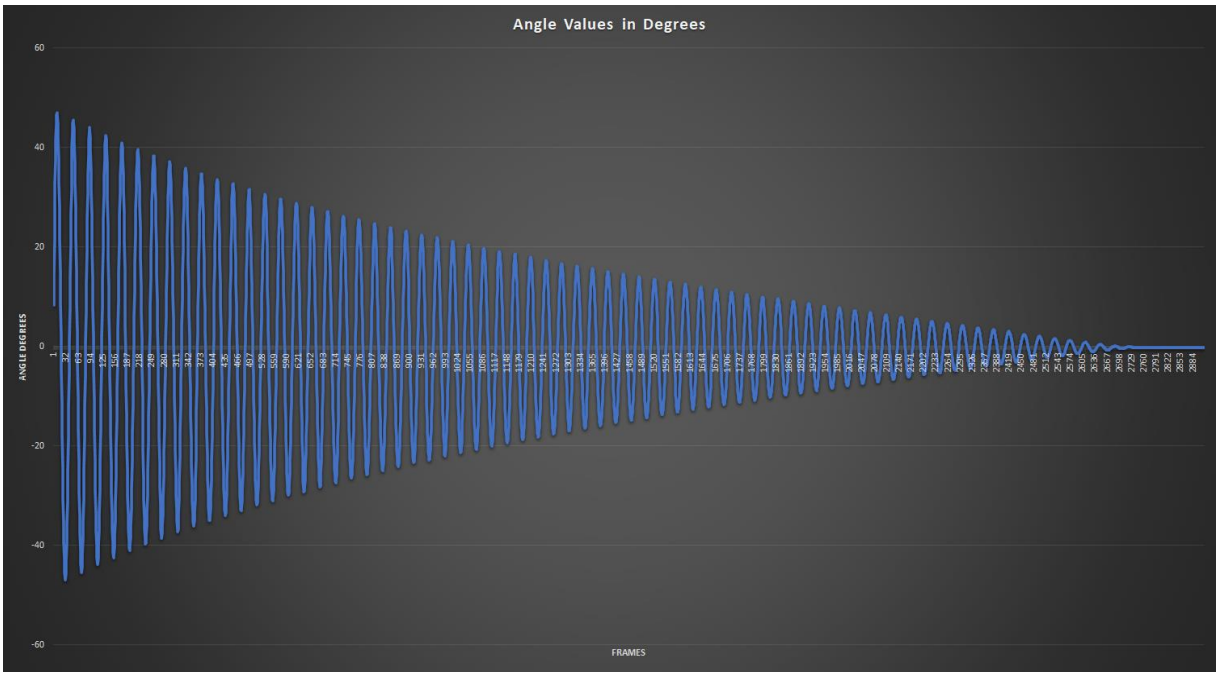

Figure 2 - Graph showing Angles in degrees over frames passed.

3.3 Task 3 – Cross Correlation

A Solution shown in video [F], As it is shown the code fully works and each component is fully cross references with the source image and shown on the PCB image as a rectangle in red.

4. Conclusion

Overall, all tasks have been successful with good results. For task 1 - Colour Sorter a slight improvement can be made following this thread on stack overflow by members of the public [6] as it shows that RGB values can be divided further into their colour range to give a more accurate output data. However, it meets the client's needs as the program solution can sort out the database of cars into coloured categories.

Task 2 - Colour Tracker has been fully successful and an optimised code has been provided to the researcher. The solution provided accomplishes the task of calculating angles from the pendulum in motion till equilibrium has reached, Angle data was then saved to a .csv file for graphing as required.

Task 3 - Cross correlation has also been a huge success, Images are cross referenced to find out any errors that might exist in the PCB board and this has been output for the user to see.

In Conclusion, it works, and the data acquired is feasible and efficient enough to do the tasks quickly.

5. References

1. RGB Colour Codes Chart (No date) *RGB Colour Chart,* Rapid Tables [Online] Available: https://www.rapidtables.com/web/color/RGB_Color.html [Accessed 19 February 2021]

2. Template Matching (No date) *Template Matching,* OpenCV, Open-Source Computer Vision, 3.4.14-pre [Online] Available: https://docs.opencv.org/3.4/de/da9/tutorial_template_matching.html [Accessed 1 March 2021].

3. Object Detection (No date) *Object Detection,* OpenCV, Open-Source Computer Vision, 4.5.2-pre [Online] Available: https://docs.opencv.org/master/df/dfb/group__imgproc__object.html#gga3a7850640f1fe1f58fe91a2d7583695da5382c8f9df87e87cf1e9f9927dc3bc31 [Accessed 19 February 2021]

4. Pillai, Anjana (2017) *Using Atan2 to find Angles in OpenCV, is there Another Method?,* Research Gate, Karunya University, 27 February [Online] Available: https://www.researchgate.net/post/Using-atan2-to-find-angles-in-openCV-is-there-another-method [Accessed 20 February 2021]

5. Color Space Conversion (No date) *Color Space Conversion,* OpenCV, Open-Source Computer Vision, 3.4.14-pre [Online] Available: https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#gga4e0972be5de079fed4e3a10e24ef5ef0aa4a7f0ecf2e94150699e48c79139ee12 [Accessed 23 February 2021]

6. Detect RGB Color Interval with OpenCV and C++ (No date) *Detect RGB Color Interval with OpenCV and C++,* Stack Overflow [Online] Available: https://stackoverflow.com/questions/9018906/detect-rgb-color-interval-with-opencv-and-c [Accessed 16 February 2021]

7. Rogers, James (No date) *OpenCV C++ Basics*, DLE University of Plymouth [Online] Available: https://dle.plymouth.ac.uk/pluginfile.php/2175642/mod_resource/content/2/OpenCV%20Basics.pdf [Accessed 27 February 2021]

6. Sources

A. VP Online - Online Drawing Tool (visual-paradigm.com) [Flowchart Task 1] [A]
B. VP Online - Online Drawing Tool (visual-paradigm.com) [Flowchart Task 2] [B]
C. VP Online - Online Drawing Tool (visual-paradigm.com) [Flowchart Task 3] [C]
D. https://youtu.be/oZsNvlrCoDM [Task 1] [D]
E. https://youtu.be/BhMQs9mAwkg [Task 2] [E]
F. https://youtu.be/VvT6nIg1OAY [Task 3] [F]

7. Revision History

Z.Shah 03/03/2021 First draft of the report