Driving With HSV

Varying Motion with Various Colors

Project Team Paul Thai, Huy Tang, William Tam, Patrick Tejada

Electrical and Computer Engineering Advisor: Professor Mohamed El-Hadedy

CalPolyPomona
College of Engineering
Electrical and Computer Engineering

Cal Poly Pomona | College of Engineering | Electrical and Computer Department

Introduction

By using OpenCV in conjunction with a Raspberry Pi 4, the team created an image processing program that can be applied to a RC car. Utilization of HSV - hue, saturation, and value, is a way to determine colors and use that data to produce outputs in order to create a self-driving car. HSV can be accurate to a specific degree which means this program can be applied to various applications depending on the user's need. In addition to creating a HSV driving vehicle, the team can determine and compare the effectiveness of the Raspberry Pi 4 in comparison to other CPUs by timing how long it would take for the program to run the color detection portion of the code. Although simple, the task aims to prove the powerful capabilities of image processing.

Problem Statement and Constraints

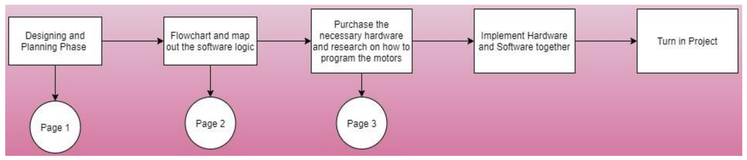

Figure 1: Autonomous Car Logic Diagram

Problem Statement:

- The aim is to utilize OpenCV and construct a color recognition program in order to apply it to an RC car

Constraints:

- Due to the ongoing pandemic, in person scheduling is limited and most work must be done remotely.
- Financial funding limits the resources provided to construct a physical vehicle

Goal:

- Utilize OpenCV to create a program that processes images using HSV
- Design and create a vehicle capable of autonomous driving using HSV

Abstract

Automation is the future. Taking a simple task and automating it alleviates the need for humans to partake in the process which allows for better allocation of time elsewhere. There are three types of automation: Process, Integration, and AI. Process automation is handled by specially defined software and applications that are tasked with that process. Integration automation allows machines to mimic human tasks once their parameters are defined. AI automation is the process where the AI learns from its previous situation and uses the data to consider its next decision. The paper talks about implementing the use of process automation which is run by python software with the OpenCV library. The purpose is to test and see if data received from the camera input can be used to program four motors to rotate in a certain direction at a certain speed. Using HSV (Hue, Saturation, Value) the camera can detect a certain range of colors defined by the user, and depending on the parameter specified, a command is passed to the motors telling it how it should move. The paper takes the scope of self-driving cars to an extremely basic level – moving forward or reverse at a certain speed with varying colors. This will help determine how effective HSV is at detecting distinct colors and how it can be used in self-driving vehicles. The paper explains how different CPUs on different OS's will affect the performance of OpenCV by using a timer that will calculate how much time is needed for the program to run.

Keywords—HSV, OpenCV, Process Automation, Integration Automation, AI Automation, CPU, Performance. (Key words)

Computer Vision and HSV

- The Computer Vision task is to:
  - **Identify** the colors of Red, Yellow, Green and Purple
  - **Omit** certain colors that was not defined by the lower and upper bounds of HSV

  This allows us the filter out unnecessary colors that would otherwise impact the performance of color identification
- **Using OpenCV**, a real-time library optimized for computer vision and hardware, we will use to gather all required data.
  - **Color Detection:** Defines the color that the camera sees
    - Upon detecting one of the defined colors, there will be a box drawn in addition to the name of the color.
  - **HSV Filter**: Filtration method that targets and isolates regions on the image that have specific colors
    - Colors have been initialized with the HSV bounds so the camera will only pick up on certain colors.
  - **Image Detection Speed**: using a 30 FPS camera we were able to get processing rates of 25-30 FPS depending on the lighting.

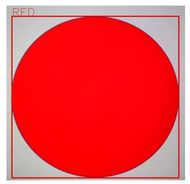

Figure 1: HSV Red

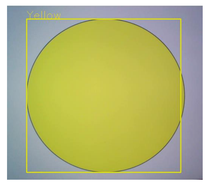

Figure 2: HSV Yellow

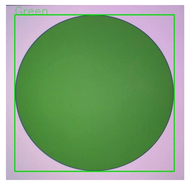

Figure 3: HSV Green

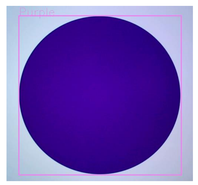

Figure 4: HSV Purple

Graphs Data

A testing set was developed using PowerPoint with timers that change slides with each having a colored circle. The testing program must detect each colored circle correctly and then finishes on the red circle which breaks the program after a few detections. For comparison, two Intel CPUs and an AMD CPU were used to benchmark as a reference to the Broadcom ARM processor on the Raspberry Pi. For each CPU, the testing data used the ten best points to eliminate outliers.

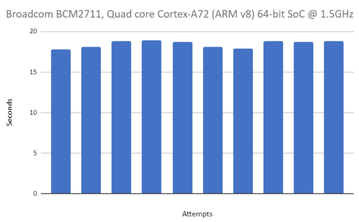

Figure 5: Benchmarking 10 runs measuring program time on a Cortex A72 (Pi)

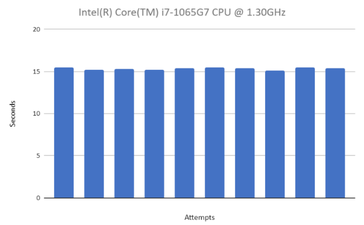

Figure 6: Benchmarking 10 runs measuring program time on an i7 7065G7

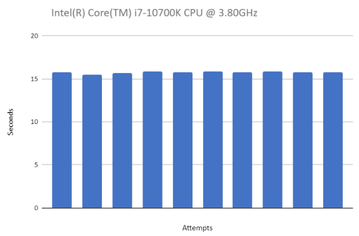

Figure 7: Benchmarking 10 runs measuring program time on an i7 10700K

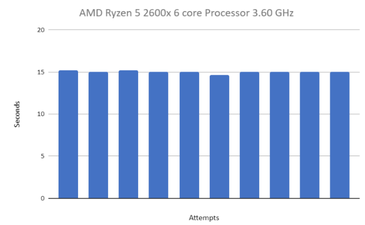

Figure 8: Benchmarking 10 runs measuring program time on a Ryzen 5 2600X

Performance and Evaluation

These results are mostly within the expected range of values. The Raspberry Pi understandably performed the worst of the testing group. It was on average around 2 to 3 seconds slower than its competitors. The averages of the others are within a one-second margin which makes sense since the CPUs are from the newer generations from both companies--10th gen Intel and 2nd gen Ryzen. There were some discrepancies in the numbers of these 3 which can be attributed to human errors and inconsistent testing methodology. This could be improved by using a modular testing system so the CPUs could be swapped out easily while maintaining the other components which should be similar to the Pi. It could be possible to eliminate sources of human error by completely automating the process of starting PowerPoint and testing at a controlled time.

Results and Impact

Purpose: Create a computer vision system using HSV colors to control the acceleration and velocity of RC motors through Raspberry Pi.

Current Design: An operational RC motor car where the camera actively inputs HSV values matching specific colors. The program interacts with the Raspberry Pi through a Linux UI where its inputs are fed by the program that calculates the HSV values of the colors received from the camera.

Design Process: The design process began with determining the HSV values that match conventional traffic lights. Afterwhich, those values are programmed onto a high-level language (Python using OpenCV) to utilize frameworks that assist in detecting the colors received by the camera. The script pipelines onto the Raspberry Pi, which receives the HSV values and runs logic statements that determine what the RC motors do after a specific value is input.

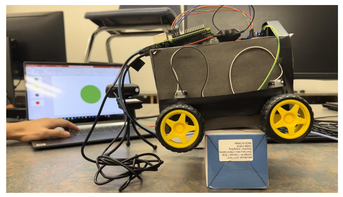

Figure 9: RC car in Working Operation

Conclusion

In this paper, we documented our work with the OpenCV library and applying for image-processing purposes. Color-detection was our main focus and the subsequent program was designed to detect primarily four colors: red, yellow, green, and purple. This program was tested and applied to control a basic motorized vehicle that behaves depending on the detected color. A basic test program was also designed to perform a basic benchmark and measure the performance of the various CPUs. Measuring the time of this program, we could make evaluations about the overall computing power of different processors on different OS. We also identified some pitfalls with our testing methodologies and ways to improve them.

Reference

[1] Nataliya Boyko; Oleg Basystiuk; Nataliya Shakhovska, "Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library," October 2018

[2] Administrator, "Raspberry Pi L298n interface tutorial: Control a DC motor with l298n and Raspberry Pi," Electronics Hub, 12-Feb-2018. [Online]. Available: https://www.electronicshub.org/raspberry-pi-l298n-interface-tutorial-control-dc-motor-l298n-raspberry-pi/. [Accessed: 23-Nov-2021].

[3] "Multiple color detection in real-time using python-opencv," GeeksforGeeks, 10-May-2020. [Online]. Available: https://www.geeksforgeeks.org/multiple-color-detection-in-real-time-using-python-opencv/. [Accessed: 23-Nov-2021].

[4] N. Congleton, "How to benchmark your linux system," Linux Tutorials - Learn Linux Configuration, 29-May-2020. [Online]. Available: https://linuxconfig.org/how-to-benchmark-your-linux-system. [Accessed: 23-Nov-2021].

[5] Gus, "How to install visual studio code for the raspberry pi," Pi My Life Up, 13-Oct-2020. [Online]. Available: https://pimylifeup.com/raspberry-pi-visual-studio-code/. [Accessed: 23-Nov-2021].

[6] Python Awesome, "A high-precision CPU and memory profiler for Python," Python Awesome, 10-Jan-2020. [Online]. Available: https://pythonawesome.com/a-high-precision-cpu-and-memory-profiler-for-python/. [Accessed: 23-Nov-2021].

[7] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The hardware/software interface