Evaluating the Performance of Hardware Using Computer Vision

Team Members:
Elena Montalvo, Dennis Lee, Rafael Hernandez, Carlos Zarco

Email:{elenar,dennisl,raloa,cdzarco}@cpp.edu

Faculty Advisor: Professor: Mohamed El-Hadedy Aly

Email:mealy@cpp.edu

California State Polytechnic University, Pomona Department of Electrical and Computer Engineering, College of Engineering

CONTENTS

Evaluating the Performance of Hardware Using Computer Vision

I. Introduction

In 2020, amid transportation restrictions and pandemic-induced delays, a shortage of Raspberry Pi devices was encountered. Consequently, the team's objective is to assess alternative microcomputers with comparable or superior capabilities to the Raspberry Pi 4. This evaluation involves a comparative analysis of the performance of the Raspberry Pi 4 Model B, Raspberry Pi 3, Le Potato by Libre Computer, and VirtualBox. The assessment is conducted by implementing Computer Vision techniques with OpenCV libraries on a Linux Operating System, specifically for color recognition in both video and image formats. The key metrics under examination include execution time, CPU usage, and memory utilization, with a focus on comprehending their impact on overall performance and overall cost analysis.

II. Hardware

A. VirtualBox

VirtualBox serves as cross-platform virtualization software, enabling users to expand the capabilities of their existing computer by concurrently running multiple operating systems, including Microsoft Windows, Mac OS X, Linux, and Oracle Solaris. The team's utilization of VirtualBox aims to standardize the performance evaluation of computer vision algorithms by configuring identical Base Memory (set at 2000MB to emulate the 2 GB memory of the Raspberry Pi 4) and employing the same quad-core CPU. This approach aligns the technological specifications of the virtual machines as closely as possible with those of the Raspberry Pi 4.

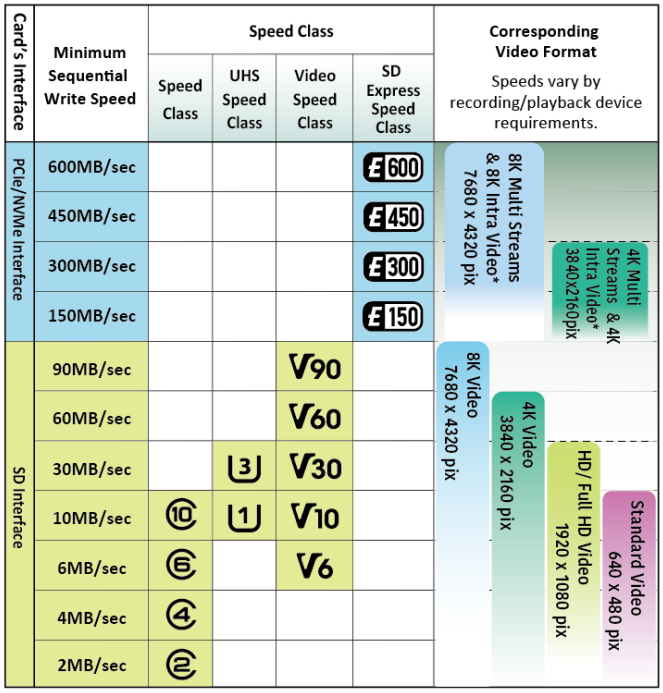
Additionally, the team ensures uniformity in the operating environment by employing the same Linux version, specifically Ubuntu 22.04.3, thereby eliminating any potential impact of the operating system version on the performance of the color detection algorithm. Four virtual machines, each configured with these specified parameters, execute the color detection algorithm to assess metrics such as execution time, CPU usage, and memory utilization.

It's noteworthy that each computer, including its virtual machine, comprises distinct internal components. The primary objective is to discern whether these variations in internal components influence algorithm performance and, if so, identify the root causes of any observed data discrepancies. The color detection algorithm is executed on each machine to analyze color in real-time through a webcam feed, identify colors within images, and ascertain color in video content.

B. Microcomputers

The team's objective is to thoroughly examine three distinct Reduced Instruction Set Computers (RISC), renowned for their implementation of a streamlined and optimized set of instructions, differing from the specialized instruction sets present in other architectures. Our analysis centers on the Raspberry Pi 4 Model B, the Raspberry Pi 3, and the Le Potato, all of which operate on 64-bit ARM microcomputer architectures. The Raspberry Pi 4 Model B is equipped with the Broadcom BCM2711 System-on-Chip (SoC), featuring a quad-core Cortex-A72 CPU clocked at 1.5 GHz. Throughout our investigation, the Raspberry Pi 4 is configured with 2GB of RAM, enhancing its computational capabilities. Concurrently, the Raspberry Pi 3 and the Le Potato adopt quad-core ARM Cortex-A53 architectures. The Raspberry Pi 3, powered by the Broadcom BCM2837, is furnished with 1GB of RAM. In contrast, the Le Potato by Libre Computer showcases a Mali-450 MP3 GPU in tandem with 2GB of RAM, contributing to its computational prowess and graphics processing capabilities.

Extra things needed to set up a desktop with the Raspberry Pi 4:



- Micro SD card
- Mouse
- Keyboard
- Desktop
- HDMI to mini HDMI
- Type C Power Cable

Raspberry Pi 3:

- Micro SD card
- Mouse
- Keyboard
- Desktop
- HDMI to HDMI
- Type micro USB power Cable

Le Potato:

- Micro SD card
- Mouse
- Keyboard
- Desktop
- HDMI to HDMI
- Type micro USB power Cable

C. SD cards

SD cards play a crucial role as the primary storage medium for microcomputers like the Raspberry Pi, encompassing the storage of essential components such as the operating system, software applications, and user data. Unlike traditional computers, the Raspberry Pi lacks built-in storage, making an SD card not just a storage solution but a fundamental component necessary for the proper functionality of the system.

During the startup process of the Raspberry Pi, the device boots directly from the SD card. When the user powers up the Raspberry Pi, the system retrieves essential files from the SD card to initiate the boot sequence. This includes the bootloader, a small piece of software responsible for launching the operating system. Both the bootloader and the operating system itself are typically stored on the SD card.

The choice of an SD card for the Raspberry Pi is a thoughtful process. It is essential to choose a reliable and appropriately sized SD card to ensure the smooth and efficient operation of the system. Take, for example, the SanDisk 32 GB ImageMate microSDXC UHS-1 Memory Card, with specifications such as Class 10 (C10) for minimum data transfer speeds, UHS Speed Class 1 (U1) for higher data rates, and compatibility for Full HD video recording. This microSD card boasts transfer speeds of up to 120MB/s, making it well-suited for applications where rapid data throughput is crucial.

The read and write speeds of the SD card directly impact the performance of the Raspberry Pi, especially in scenarios where data access speed is a critical factor. Therefore, the careful selection of an SD card with favorable read and write speeds is an important consideration for users aiming to achieve optimal performance in their Raspberry Pi-based projects.

III. Result

A. Test performed

The team develops three Python scripts that measure the performance of the hardware running the algorithm for color detection on a video, picture, and live camera feed from the webcam. To accomplish this, the team uses using the following Python Libraries: `psutil` and `time`. `psutil` stands for **p**rocess and **s**ystem **util**ities. The team utilizes this library to retrieve information on memory and CPU usage for the virtual machines and Raspberry Pi 4. The team measures this usage as a percentage of total memory/CPU for each of the 10 times the team runs the algorithm. The time library is used for time-related operations. In the implementation, the team uses it to measure how long the algorithm

takes to execute in seconds. For the two videos tested, the team also uses the library to ensure that the algorithm is running for at least 30 seconds on the videos. This decision ensures enough strain on the chips to have measurable results. The team is also implementing the code in this way to easily run multiple tests. The 30-second cap helps with testing since the videos are longer than this duration, it makes testing a faster process. Also, the execution time is often over 30 seconds due to the time it takes to execute and process the code itself. These deviations reveal important information about the performance of the hardware. All tests are done with machines connected to power since the laptops where the VMs are running could throttle under battery power or battery saver modes.

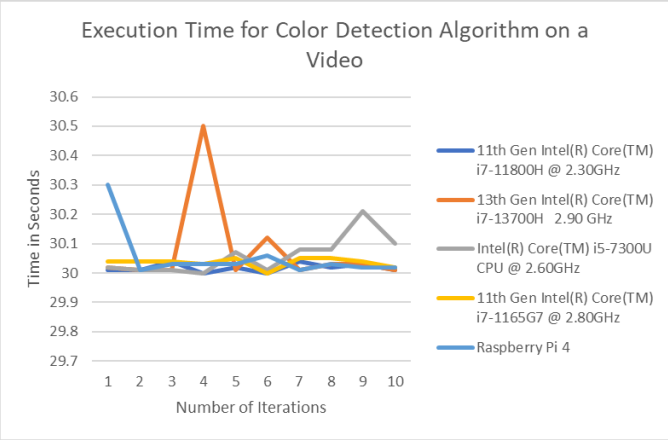


Fig. 1: Execution Time for Color Detection Algorithm on a Video

B. Picture

The team manages to get the picture script to function correctly on the VMs, however, it was the last of the tests. By this point, the team experiences problems with the Raspberry Pi 4. When it is plugged into power, and hooked up to an external display, keyboard, and mouse there is no video signal whatsoever so all the team sees is a black screen. This issue does not occur for the video or the webcam tests. The Raspberry Pi 4 is functioning correctly at that time so the team measures data in full for the other scripts. Unfortunately for the picture, the team cannot test the Raspberry Pi 4. The team is not particularly sure about what caused this sudden issue either; the Pi was fully functioning and no changes were made to the SD card which has the OS loaded onto it. The team tries removing the SD card and then plugging it in to get the bootloader to pop up on the screen but that also proves to fail. It appears that the microcomputer also does not have a bootloader now, and even though the SD card is formatted correctly and has Linux installed on it the team is not able to perform the test for the color detection on a picture for the Raspberry Pi 4 that is being used.

C. Video 1

The Raspberry Pi 4 compares very well to all 4 of the VMs in every category measured. The execution time is consistently near 30.0 seconds and for all 10 iterations averages to about 30.054 seconds. Which is only the 3rd lowest of all the tested devices. This is impressive considering the weaker processor of the Raspberry Pi 4. The 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz has the best execution time on average with only 30.018 seconds. The 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz follows with an average of 30.036 seconds. The execution time amongst all devices however is relatively consistent and stable around the 30.0 second mark. While the Raspberry Pi 4 is not the strongest performer, it still holds its own and puts up respectable numbers. There does appear to be one outlier in the data: the 13th Gen Intel(R) Core(TM) i7-13700H 2.90 GHz has one iteration where the execution time comes in at 30.5 seconds, which is the highest of all the devices. This appears to be an outlier because it is difficult to replicate, and may be caused by host machine load. Perhaps a background process in Windows 11 which is the native OS of the machine causes a sudden spike. If this iteration is ignored, the average execution time for this machine drops significantly and becomes the 2nd fastest machine of the 5 tested here. This is more in line with what the team expects since this machine has the most capable chip.

The CPU usage data for the first video reveals that there are slight changes in performance between running the algorithm on a virtualization layer and natively on the hardware itself. The Raspberry Pi 4's 4 Core ARMv8 Cortex-A72 1.8GHz performs the best of all the devices, boasting the lowest average CPU usage of 0.76%. This is followed by the 13th Gen i7 processor at 1.52% and lastly, the 11th Gen Intel i7-1165G7 @ 2.80GHz performs

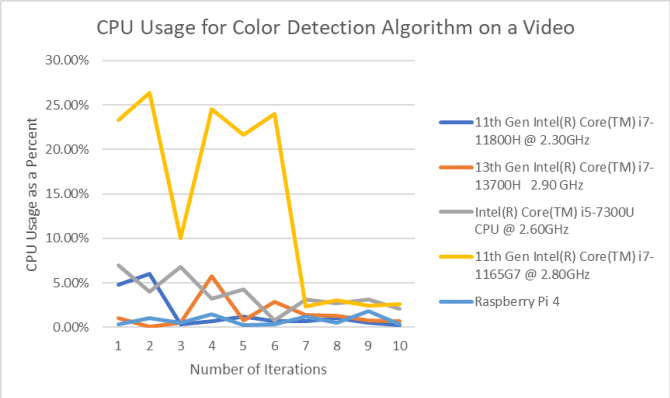


Fig. 2: CPU Usage for Color Detection Algorithm on a Video

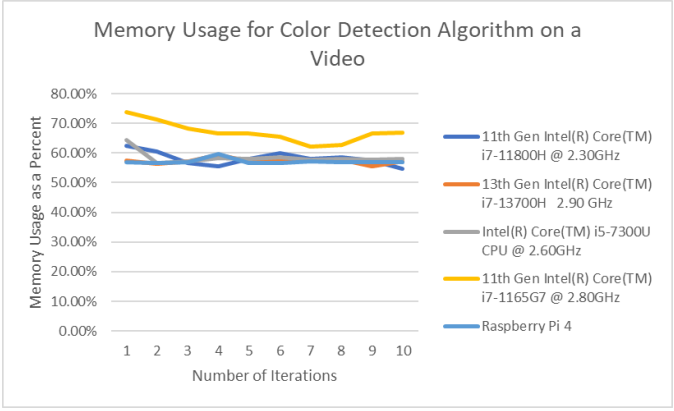


Fig. 3: Memory Usage for Color Detection Algorithm on a Video

the worst with the highest CPU usage of 14.01 %. However, it is important to note that there are large spikes in CPU usage for this processor as well. This issue is replicated, but the latter half of the tests for this processor are more in line with all the other devices. It is highly likely that there is a background process impeding the performance and that it finishes halfway through the tests since there are such large differences between the CPU Usage from the beginning to the end. So assuming something impeded the quality of the data for this device, does it change how well the Raspberry Pi 4 does compared to the VMs? The answer is no. The Raspberry Pi is also easily outperforming the other 11th Gen Processor used in our tests as well as the Intel(R) Core(TM) i5-7300U CPU @ 2.60GHz.

The Memory Usage data also seems to demonstrate similar conclusions that are made when looking at the CPU usage. Similarly, the Raspberry Pi 4's average RAM usage is the lowest at 57.19% with the 13th Gen i7 machine following closely at 57.25%. The 11th Gen Intel i7-1165G7 @ 2.80GHz performs the worst here as well with the highest spike in ten tests being at 73.80%. This is a significant increase, however, the team speculates the cause of such stark differences being a possible background process terminating mid-test. This graph further illustrates that this may be the cause of higher resource usage across the board for this machine since the latter iterations show a significant drop to a low of 62.30%. However, this is still higher than most machines resulting in worse performance across the board. Most machines perform worse than the Raspberry Pi 4 but the more powerful machines are the nearest in terms of memory consumption.

D. Video 2

There is a significant difference between the two videos tested. Video 1 shows solid colors that fill the whole screen and rapidly change. In contrast, in the second clip, various colors are coming from different edges of the screen colliding with each other rapidly. The smokey look of the colors in this video leads them to mix. The additional colors in the frame mean that the algorithm may have a harder time differentiating between the color it is identifying and the others. When looking at the execution time for this video, all machines perform similarly with the lowest average execution time going to the 13th Gen i7 processor at 30.014, and the lowest being the Intel(R) Core(TM) i5-7300U CPU @ 2.60GHz at 30.046 The difference between these is so minuscule that they are within fractions of a second. The execution time is the most stable metric for both video tests and there are no significant differences between the Raspberry Pi 4 and the VMs when running the Color Detection Algorithm. Although the VMs tend to have higher spikes in execution time than when running natively on the hardware like with the Raspberry Pi 4, the differences are not great enough to draw significant conclusions about the differences in performance. VMs may have spikes because of emulation, the load of the host PC, and limitations in virtualization technology.

CPU usage for Video 2 is far more interesting. For simplification purposes, the team argues that

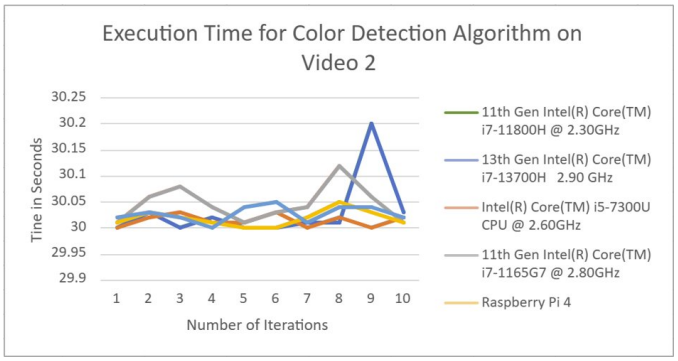


Fig. 4: Execution Time for Color Detection Algorithm on Video 2

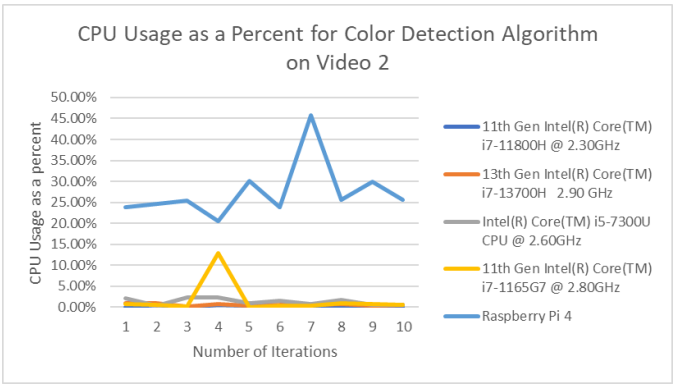


Fig. 5: CPU Usage as a Percent for Color Detection Algorithm on Video 2

all VMs it tests perform similarly with CPU usage being consistently less than 2%. However in stark contrast, the Raspberry Pi 4's 4 Core ARMv8 Cortex-A72 1.8GHz performs significantly worse with the highest CPU usage of all the machines reaching a high of 45.70% and averaging at about 27.53%. This is a drastic change observed during testing that indicates that the type of video chosen for the algorithm to analyze greatly affects the resource usage for processing on the Raspberry Pi's ARMv8 Cortex-A72 processor. It is important to acknowledge that the high of over 45% is likely an outlier as it only occurs once however the rest of the data is between about 23-30%. This change highlights the inaccuracies of emulating hardware and the limitations that virtualization software can have with accurately representing the hardware that is emulated. The Raspberry Pi suffers significant performance degradation compared to the VMs but it likely more accurately showcases the true nature of the intensity of the algorithm itself and how it would run on similar hardware. Unfortunately since the team could not get Le Potato with its 4 Core ARMv8 Cortex-A53 1.512GHz processor to load and display its OS properly, it can only speculate as to what causes such big differences in performance and if it would be similar on this processor as well.

The memory usage for the Raspberry Pi 4 is also the worst with an average consumption of 77.08%. A large change from the 57.19% average for video 1. Video 2 proves to be more intensive on the system as a whole for every metric but execution time, where the Raspberry Pi 4 manages to stay within range of all the VMs. The difference in RAM usage is large here but not as significantly large as the differences in CPU usage for this test. The

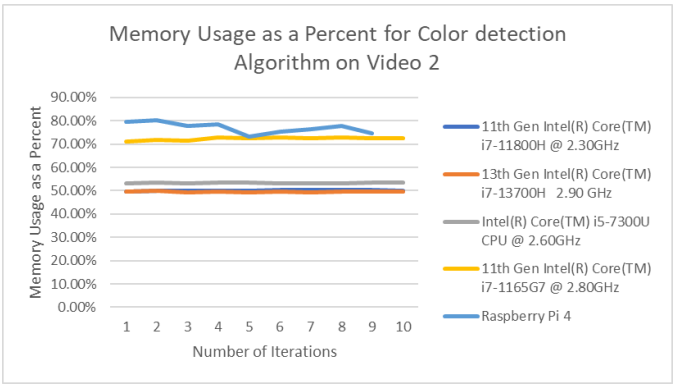


Fig. 6: Memory Usage as a Percent for Color detection Algorithm on Video 2

laptop whose performance most closely mirrors the Pi here is the machine with the 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz which manages an average CPU usage of 72%. All other systems are approximately between 50-53% CPU usage. The Raspberry Pi 4 more accurately depicts the performance on real hardware and all VMs even manage to have lower RAM usage as they do when running the same algorithm on Video 1.

E. Webcam

Running the color recognition algorithm through a live feed from the devices' webcams is one of the most interesting parts of this project. While the devices ensure that the program is running, they also must focus on running their webcams throughout the program. This task requires the processors to put extra effort into running this program compared to the other programs. This effort might be visible in the results. This part of the project does not use the Intel(R) Core(TM) i5-7300 CPU 2.60 GHz because

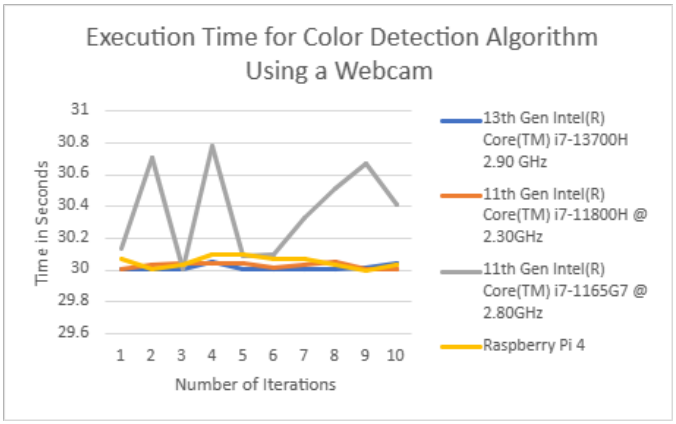


Fig. 7: Execution Time for Color Detection Algorithm Using a Webcam

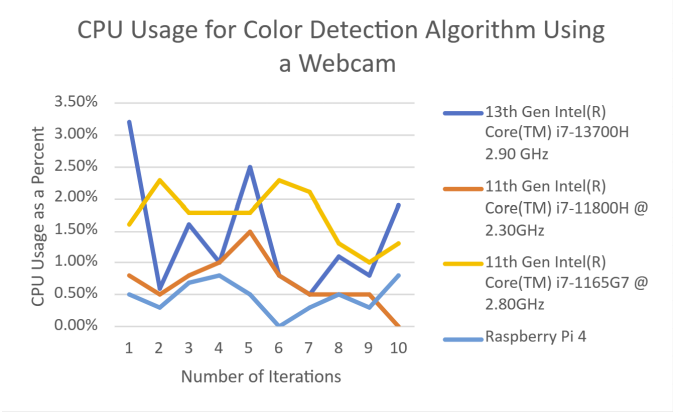


Fig. 8: CPU Usage for Color Detection Algorithm Using a Webcam

the device's webcam couldn't function. With all this in mind, the team will look at the results from this section of the experiment.

From the execution time results, the team can see that most devices stay consistently between 30 to 30.07 seconds of runtime for this program. The only exception to this concept is the 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 GHz. Its execution time varies dramatically through all ten iterations of the program. This processor spends an average of 30.375 seconds per iteration. It spends more time on this program than any other device. It takes a maximum of 30.78 seconds to run this program. The lowest runtime for this program is 30.00 seconds on the Raspberry Pi 4, and the lowest average runtime is 30.018 seconds from the 13th Gen Intel(R) Core (TM) i7-13700H 2.90 GHz. The abnormal fluctuation of the execution time for the 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 GHz can be due to the processes running in the background. The strain from the webcam can also cause these inconsistent run time results. Depending on how much feed the processor decides to process, the program can help take up more or less time for each separate iteration. In the case of this processor, it might have to work more than usual to run the webcam, which, in turn, requires an inconsistent amount of time each run to finish the program. Overall, most of the virtual machines compare well to reflect how the hardware of the Raspberry Pi would be in terms of execution time for this webcam program.

From the perspective of CPU usage, there seems to be no trend on how much CPU the processors use. The range of CPU usage values obtained seems abnormal as well. The team gets values as low as 0.00% in some iterations from the Raspberry Pi 4 and the 11th Gen Intel(R) Core(TM) i7-11800H 2.30 GHz. This value could mean that less than 0.005% of the CPU runs this program in these instances. The team believes that it only shows 0.00% due to some precision limitations. The highest value is 3.20% in the first run with the 13th Gen Intel(R) Core(TM) i7-13700H 2.90 GHz. The best average CPU is an average of 0.47% from the Raspberry Pi 4. On the contrary, the worst average comes from the 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 GHz with a value of 1.73%. In terms of CPU usage, the Raspberry Pi outperforms the virtual machines. This discrepancy could be because background processes from the processor simulating the Raspberry Pi taking up parts of the CPU. Despite this, there is no processor that can fully replicate the result we have gotten from the Pi very accurately.

Despite the unpredictable results previously mentioned, the memory usage stays consistent with each iteration. Each processor keeps a consistent array of memory usage values. From the results, the lowest memory-consuming processor for this webcam program is the 11th Gen Intel(R) Core(TM) i7-11800H 2.30 GHz. This processor gets an average of 50.34% memory usage and produces the lowest memory usage value from this program. The lowest amount of memory used for this program is 50.00%. The 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 GHz takes the most memory to run the program. It achieves a maximum of 69.40% memory usage to run the program and an average of 68.43%. Each

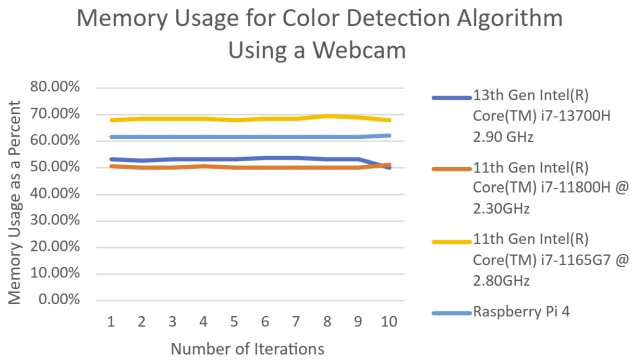


Fig. 9: Memory Usage for Color Detection Algorithm Using a Webcam

processor has different ranges, and they maintain within these ranges. Since the team is not relying on media stored in the devices, the memory should not fluctuate as much as it does with the other experiments. Comparing the processor to the Raspberry Pi, none of the processors can simulate the exact same range of memory usage as the Pi did, but some processors can do better than the Pi in this aspect.

IV. CONCLUSION

There seems to be no direct substitute for the Raspberry Pi. None of the virtual machines can guarantee similar values to the Raspberry Pi on all tests. If needing to simulate a Raspberry Pi 4's performance on a color recognition algorithm, a processor that will operate closely to the Pi based on how one plans to run the algorithm has to be chosen. To simulate a Raspberry Pi when detecting color in a simple video, use the 11th Gen Intel(R) Core(TM) i7-11800H 2.30 GHz. This processor would also be good to get realistic values of a Raspberry Pi 4's CPU usage and execution time, except for the CPU usage when using the algorithm for a complex video. To accurately simulate the color-detecting algorithm running on a Raspberry Pi 4's memory with a more complicated video or webcam, there is no processor that can perform this task. No processor is suitable for measuring CPU usage during complex video playback. Overall, to use simulators to measure the performance of a Raspberry Pi 4, one must use processors that would best reflect the aspect of the hardware to measure.