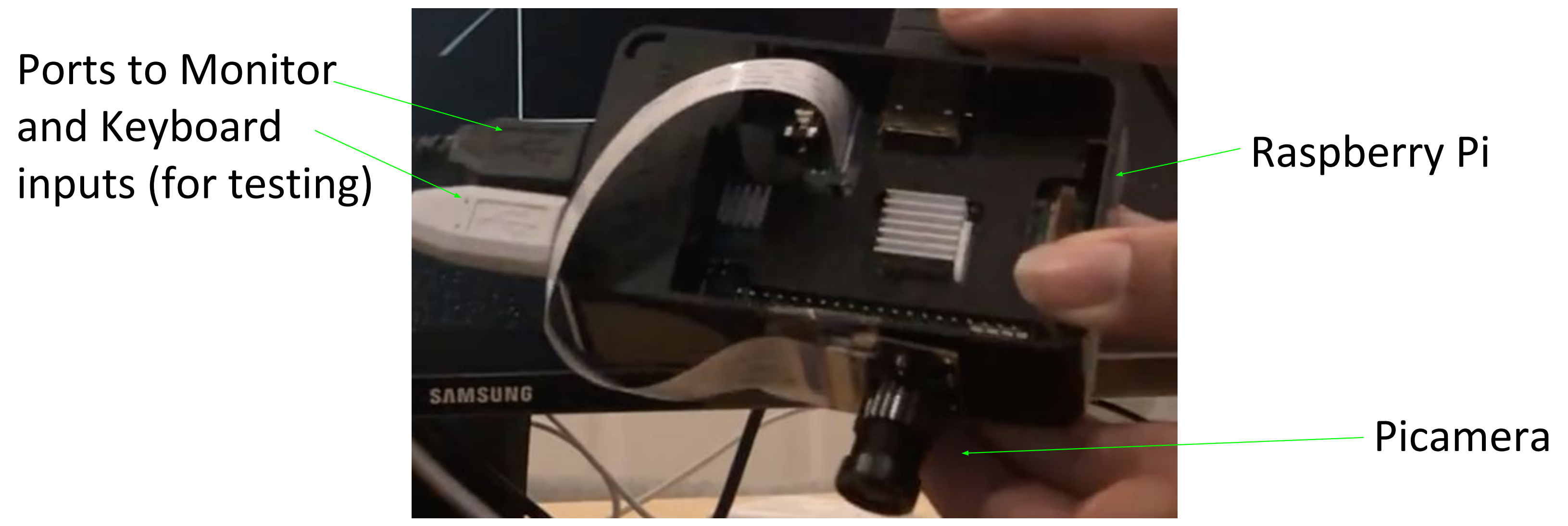
Is It Feasible to Build a Cheap Portable Warning System?

Akul Datta[1], Kyle Bordeau[2]
Henry M. Gunn High School[1], NASA[2]

INTRODUCTION

Automotive accidents are an ongoing epidemic in the United States. Pedestrian and total death rates from those accidents are increasing 6-11% year over year (Fatality Facts, 2017). Autonomous cars are the most promising solution to this problem; however, it will take a long time until all vehicles on the street are autonomous. For this reason, I propose a new solution: to build an affordable portable warning system (PWS) for pedestrians to reduce car crashes. My research project is to determine the feasibility of building such a device using today's latest technology in the fields of machine learning, object detection, and image processing.



Picture of my system

RESEARCH METHODOLOGIES

Before I started building the prototype, I did research on what has already been done in the field of image processing and object detection. I used that information to build the first sketch of how my system would look and work.

Figure 1: Flowchart of My Project

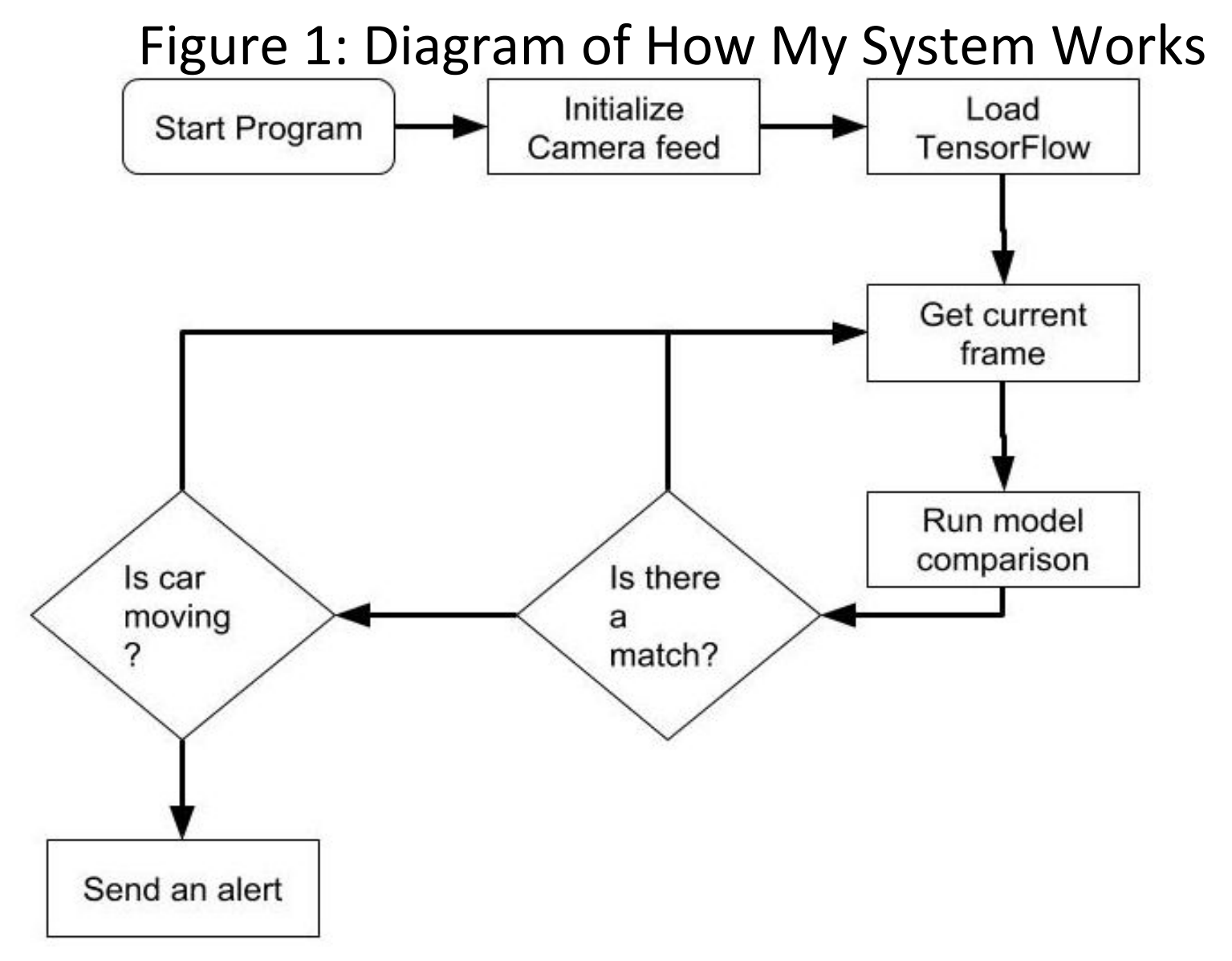


Figure 1 shows in detail how my system works. Frames in a live video feed are analyzed based on similarities to objects such as cars. If the program detects a car, it also detects its motion.

For my project, the accuracy of detection is key. Since I have proposed a pocket-sized warning system, though, the computer has a lower processing speed, and therefore cannot keep up with the highest-accuracy detectors. Fortunately, I have another lever: the complexity of my algorithms. Generally as the complexity increases, the accuracy of detection increases and vice versa. I tuned the algorithm to reduce the complexity but measure the accuracy trade off in the process. The greatest challenge is that the technology in this field is evolving and changing quickly. New frameworks are being built currently which are more efficient than previous ones. This means that this field is more dynamic than other fields of warning systems, such as the brakes on a car or traffic lights.

DATA AND FINDINGS

As I implemented a prototype for the Raspberry Pi, I collected data on the prototype to validate its feasibility. I experimented with lowering the pixel count of the video feed to see if the speed increased. When I did this, I found that even a huge difference in video size does not speed up the program by much. From this, I found that the size of the video feed has little effect on the algorithm's speed.

Figure 2: Time Comparison Between Parts of the Program

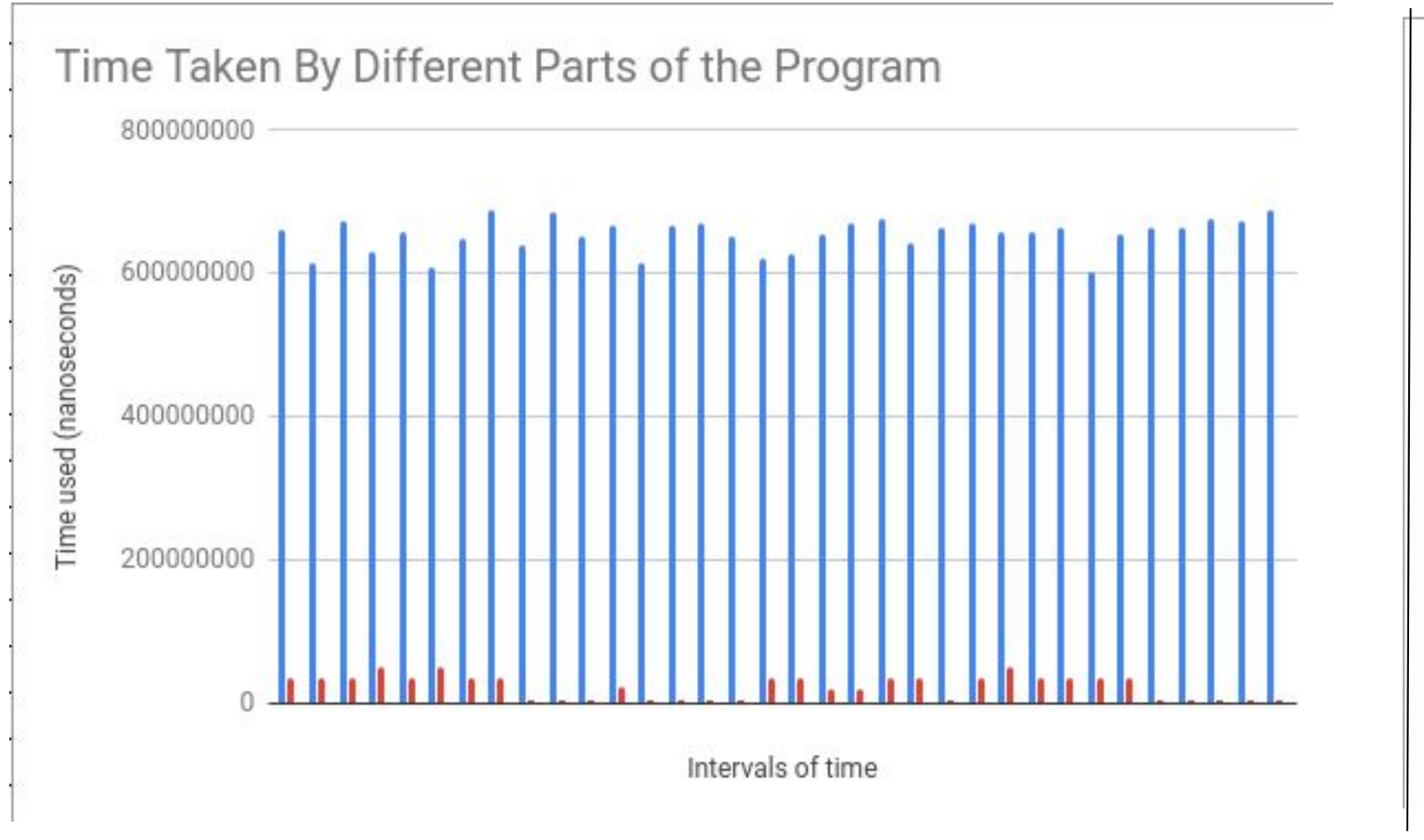
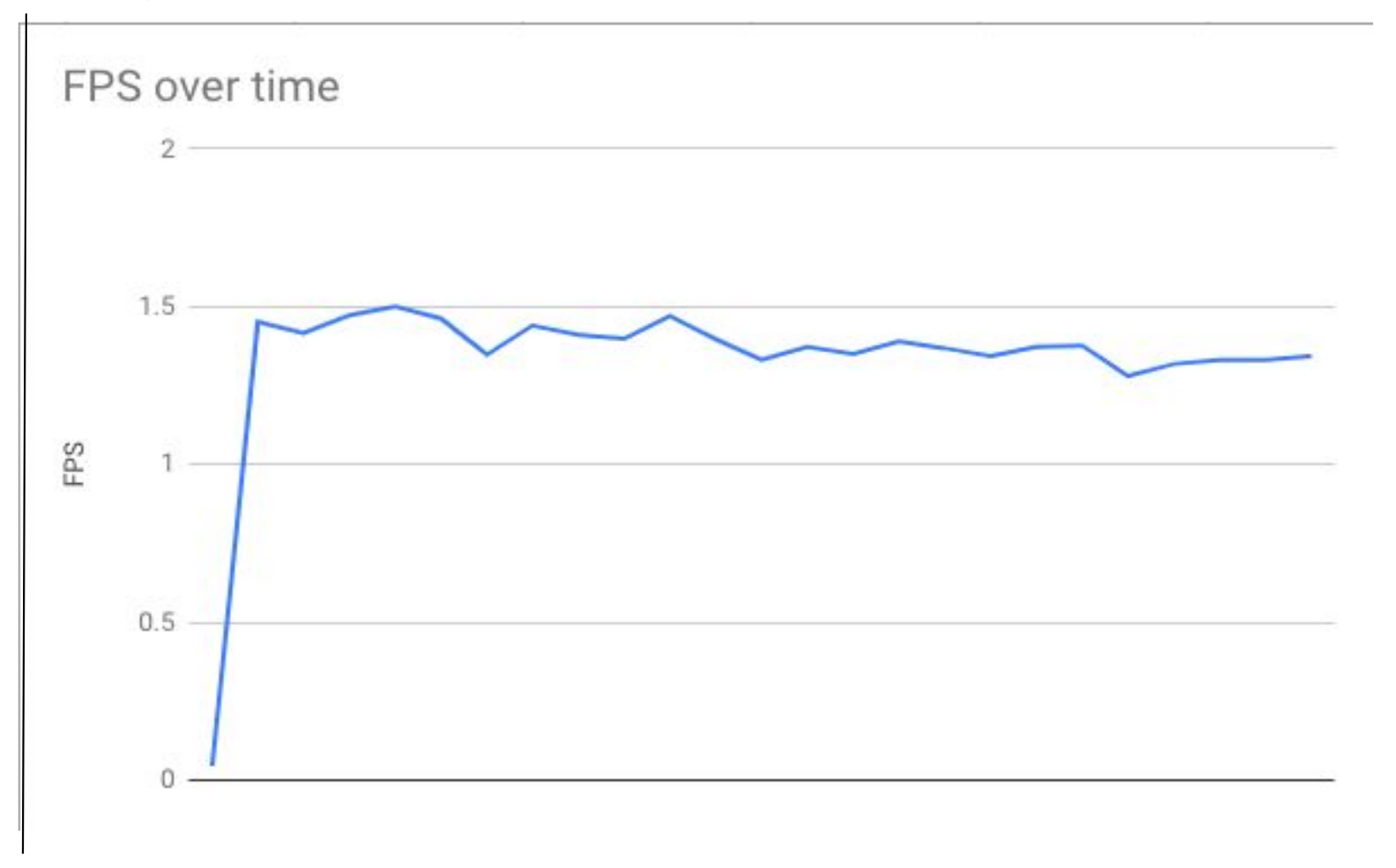


Figure 3: Frames Per Second (FPS) over Time



As shown in Figure 2, the TensorFlow algorithm (shown in blue) takes up 95% of the total processing power. Changing the algorithm is beyond the scope of this project. The other 5% is minimal and will not affect the CPU usage or frame rate much.

Figure 3 shows the current frame rate of my project in frames per second (FPS). The target to build a viable system for usage needs to be at least 5 FPS, but, I could only get an average of 1.5 due to CPU-intensive modules to run object and motion detection.

After additional research, I found that we can do a crude/basic motion detection by finding the difference between frames of videos. The method is called root mean squared (RMS), and it provides a method to run the algorithm based upon an open source project. As of now, the program works as such: if someone moves in front of the camera, the program detects motion.

CONCLUSIONS AND ANALYSIS

I conclude that I can build a system that is portable using open source frameworks and credit card sized hardware. However, due to limitations in the cheaper hardware and complexity of algorithms, we need a more powerful system with better CPU and memory, which will be more expensive. Given the rate of the current innovation, I expect that we should be able to build an affordable system in a few years.

Figure 4: Trendline of Future Enhancements

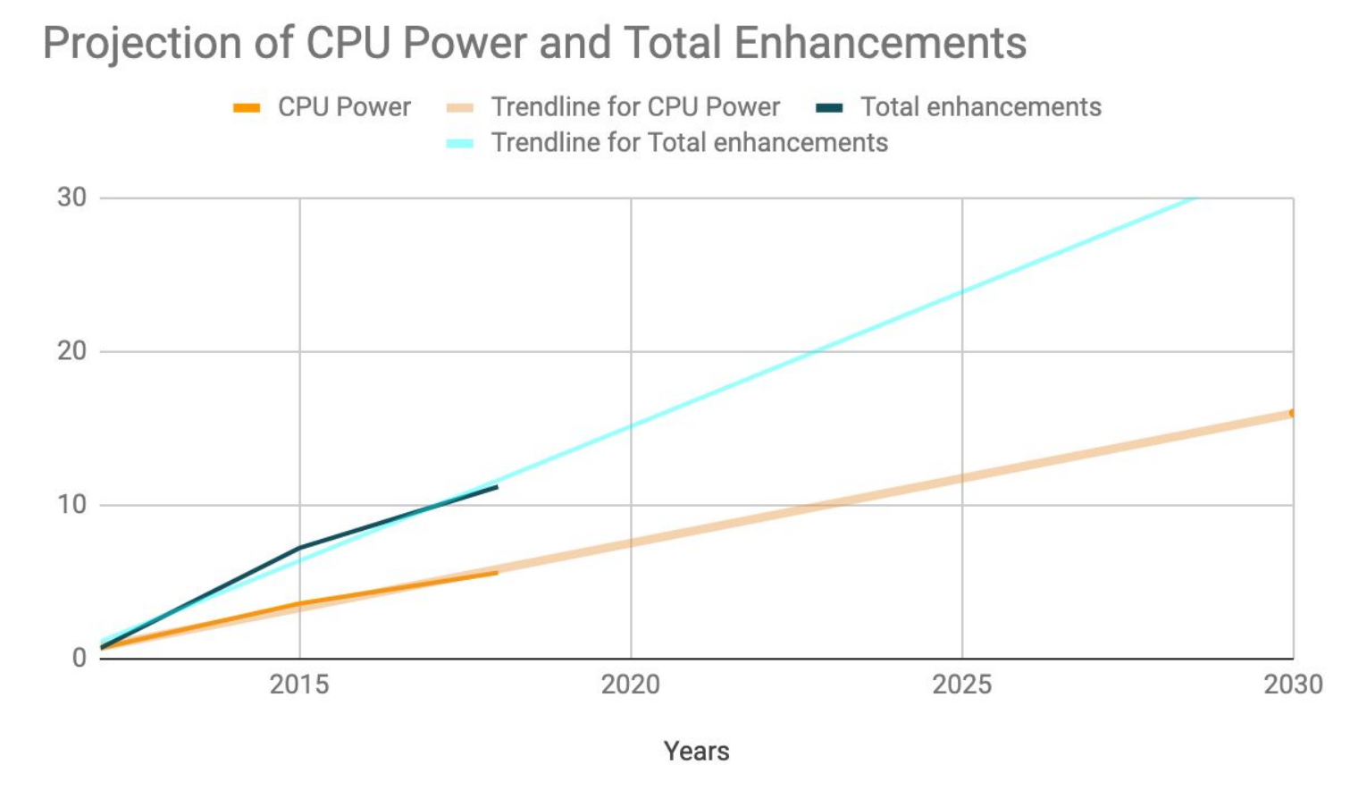
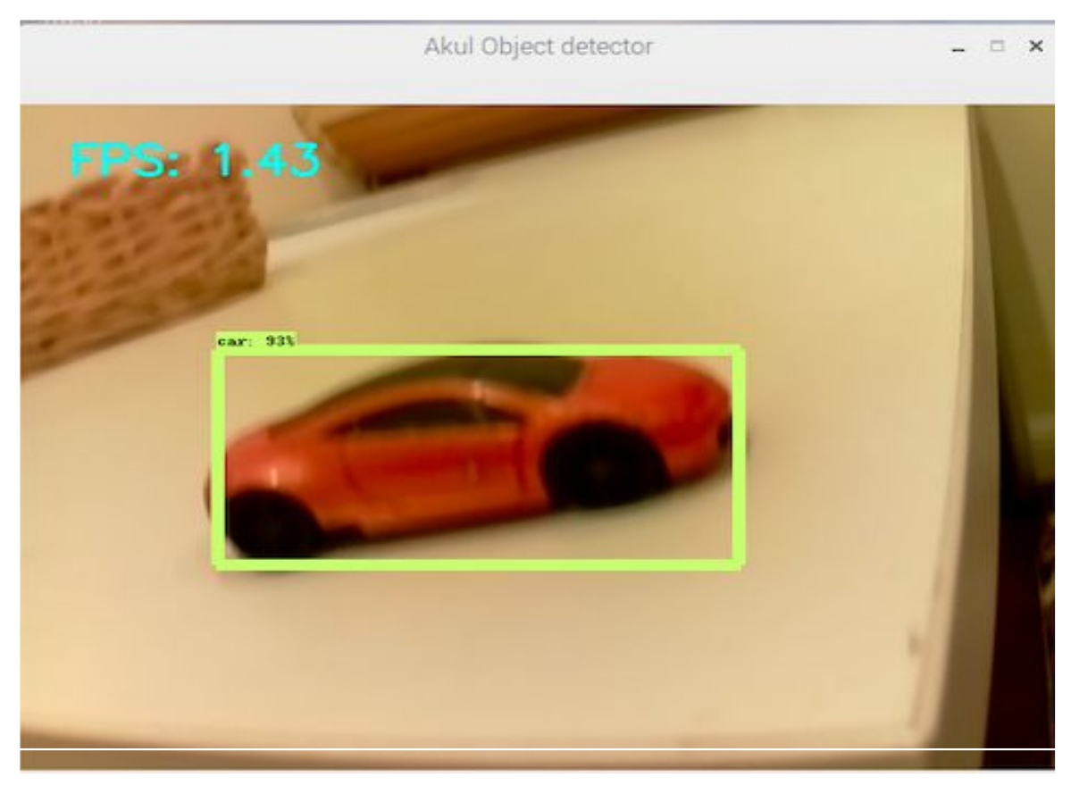


Figure 4 shows two lines, one that shows the improvement of Raspberry Pis over the last few years. That first line has a trendline that shows how much CPU power the machine has. But, the software will be advancing just as much as the hardware. Therefore, I created the "Total Enhancements" trendline, which represents the combined growth of both hardware and software in the future.

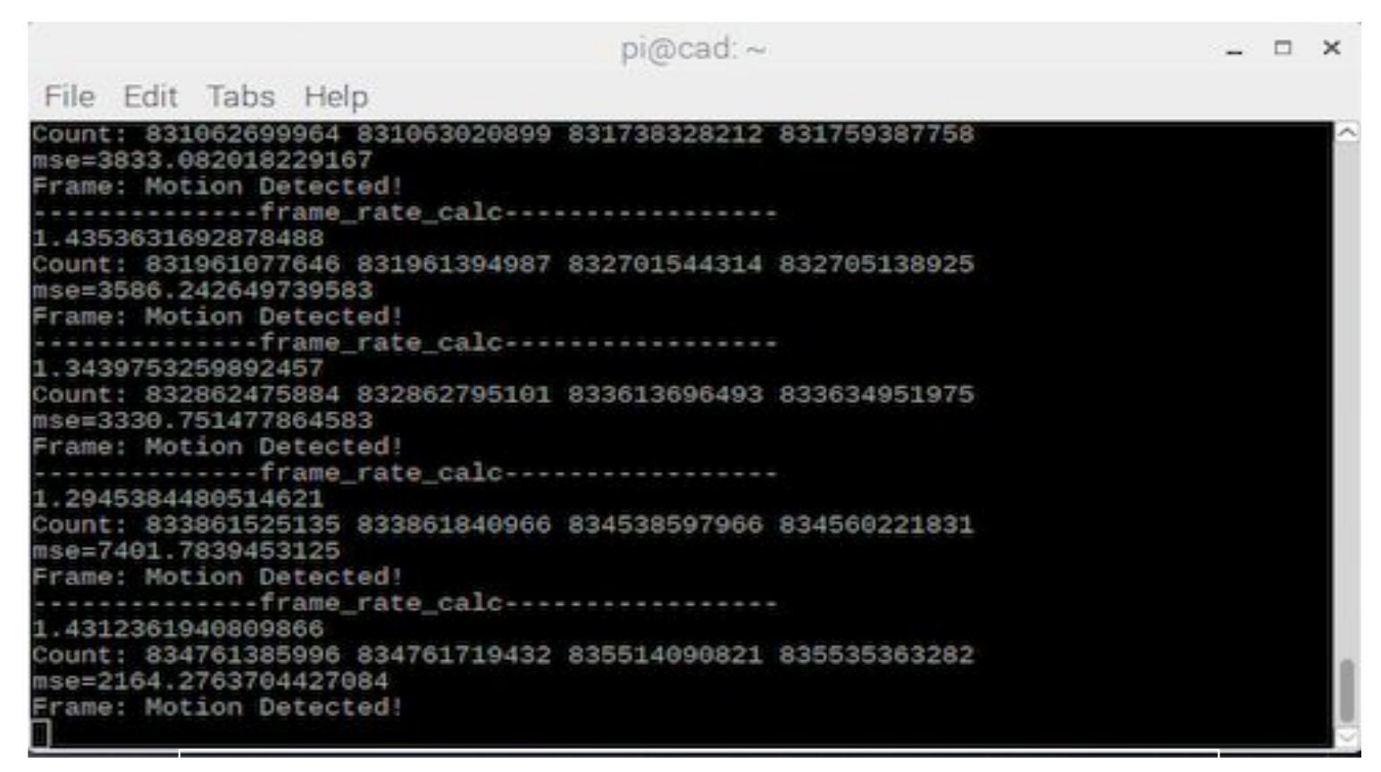
IMPLICATIONS AND NEXT STEPS

In my project, I focused on implementing the most difficult parts of the PWS. I implemented motion detection and object detection on a computer as small as a Raspberry Pi for $55. This is a feat since other sensors used for these same purposes are large and cost upwards of a thousand dollars. Due to this, there are many next steps I need to take to make my vision of a PWS a reality. I have listed a few enhancements below to improve my product:

1. **Multiple Cameras:** Probably the most important next step. The major limitation to my PWS is that it can only "see" in one direction. To give the PWS more information to protect the user from all angles, it would be better to build a system that takes input from multiple cameras so the cameras can point different directions.
2. **Add Other Sensors:** I can make the system more reliable by adding input from other sensors such as LIDAR (a sensor using light waves to give the PWS a version of the world around it) and RADAR (a sensor using radio waves to detect motion) so it can work in rain, lower visibility situations, and corner crosswalks.
3. **Other Machine Learning Frameworks:** I used the machine learning module TensorFlow for this project. There are other open source frameworks that can be tested for efficiency, speed, and CPU usage.
4. **Test for Corner Cases:** I built the system to do basic motion detection, and tested on mostly indoor objects. I need to test my system on more outdoor test cases and in all practical scenarios, such as rain, fog, curved roads, etc.
5. **Battery Support**: This system will need a battery to work. So, I need to test different types of battery packs and figure out the characteristics of power consumption.



Monitor Output for my program



```
Count: 831062699964 831063020899 831738328212 831759387758
mse=3833.082018229167
Frame: Motion Detected!
--------------frame_rate_calc----------------
1.4353631692878488
Count: 831961077646 831961394987 832701544314 832705138925
mse=3586.242649739583
Frame: Motion Detected!
--------------frame_rate_calc----------------
1.3439753259892457
Count: 832862475884 832862795101 833613696493 833634951975
mse=3330.751477864583
Frame: Motion Detected!
--------------frame_rate_calc----------------
1.2945384480514621
Count: 833861525135 833861840966 834538597966 834560221831
mse=7401.7839453125
Frame: Motion Detected!
--------------frame_rate_calc----------------
1.4312361940809866
Count: 834761385996 834761719432 835514090821 835535363282
mse=2164.2763704427084
Frame: Motion Detected!
```

Terminal Output for my program

ACKNOWLEDGEMENTS / REFERENCES

*** Special thanks to my mentor, Kyle Bordeau, for making this project possible.

***Works Cited:

Harkey, D. (n.d.). General statistics. Retrieved from https://www.iihs.org/iihs/topics/t/general-statistics/fatalityfacts/overview-of-fatality-facts

Bristeau, P., Callou, F., Vissière, D., & Petit, N. (2011, September 2). The Navigation and Control technology inside the AR.Drone micro UAV. Retrieved October 6, 2018.

Fast algorithm for camera motion detection (2013). Retrieved October 7, 2018, from https://stackoverflow.com/questions/4393841/fast-very-lightweight-algorithm-for-camera-motion-detection

Autonomous Vehicle Technology. (n.d.). Retrieved from https://books.google.com/books?id=y0WrAgAAQBAJ&lpg=PP1&ots=-7Hg81FBWN&dq=autonomous car technology&lr&pg=PR11#v=onepage&q=autonomous car technology&f=false