Effectiveness of PiCamera Motion Detection

ZACH PAUL, JIM COLLADO, KALEB PIERCE, AND JACK HAMMER

ABSTRACT

Our paper describes the experiments we undertook to determine how effective the Raspberry Pi's picamera motion detection library is at detecting motion in different environments and offers what could be the start for a home surveillance system. We describe the implementation of the code used from picamera's documentation and our methods of testing the camera's ability to detect motion. Further, we explain how we turned our findings into quantitative data that we could use to compare the performance of the motion detection system in the different environments we had chosen to test. We then explain what results we obtained after testing and which environment seemed to work the best for our motion detection system. Our results of our experiments suggest that the amount of motion detected in a low-lit environment was quite less than in standard lighting while environments with a dynamic background had returned results that were identical to what was obtained in standard lighting environments.

INTRODUCTION

A raspberry Pi is a small single board computer that was originally developed in the United Kingdom with the original intention of encouraging an education in computer science skill for both students and teachers. Since its first implementation, it has become widely used in 1000's of different ways in computer science, not the least of which is the implementation of the Picamera. A *picamera* [1] is a raspberry pi compatible module that can be used to take a high definition picture. For our research purposes, we decided to test its effectiveness in varying amounts of light. Specifically, we tested the picamera's ability to detect motion in three different, unique amounts of light. This research has many potential possible benefits that could be used in a wide array of fields. The most prevalent possible use that we found was the potential of the picamera for home security. Through our research, we will show how varying degrees of light will affect the effectiveness of the picamera and how these changes will change the picamera's ability to detect motion. Not only that, but we will also show how different types of dynamic backgrounds (that is, backgrounds that are consistently changing) can also affect the degree of accuracy from the picamera.

RELATED WORK

As revolutionary as our research was, we are not the first group to take an interest in the effectiveness of the picamera. We were hardly the first to take an interest in the picamera and its effectiveness. One project that went by the name of the *blind hat project* [2] also used the motion detection of the picamera module, this time to aid blind people in their struggles. In addition to this research project, there are also many implementations of picamera into home security. As seen in our experimental results, we can show that Picamera has fairly efficient range of motion detection, something that is obviously quite important in the field of home security. What makes picamera motion detectors a viable option for home security is the simplicity with which they can be installed, and at low cost. One thing that is for sure is that there are many different paths for picamera motion detection research to be taken in.

Author's address: Zach Paul, Jim Collado, Kaleb Pierce, and Jack Hammer.

2022. Manuscript submitted to ACM

MATERIALS AND METHODOLOGY

Fortunately for us, most of the work done for this project was completed via the Raspberry Pi, so we didn't have too many materials for our experiment. On top of the Pi, we were also given a no IR camera by Dr. Krupp, which is what we used to actually record our videos for analysis. We directly connected the camera to the Pi. Finally, we utilized a roll of duct tape rolling down an incline, the use of which will be explained later. Figure 1 shows a picture of our setup constructed in the Robotics Lab:

Fig. 1. Our setup in the robotics lab.

To reiterate, the purpose of our research was to compare how well PiCamera detects motion in differing environments. If we break this apart, we are left with a dependent, independent, and control variable. The dependent variable would be the amount of motion that was detected, while the independent would be the setting in which the video was recorded in. To elaborate, our group manipulated the robotics lab to appropriately fit the desired environments needed for our experiment. The control variable, on the other hand, was the actual motion itself. Since we are measuring effectiveness of motion detection in different envirnments, we needed only the environments to change, not the actual motion. Thus, the motion for each trial video needed to be practically identical. Achieving this was not easy. Our initial approach to replicating motion was to have one of our group members walk in front of the camera with the *hope* that they managed to be the same distance from the camera and walk the same speed for each video. However, this approach was flawed, as there seemed to be too many inconsistencies with the data. Thus, we needed a way to be sure that the motion for each video was truly identical. Thus, we came to the conclusion that we would have an object (duct tape) roll down an incline in front of the camera. By doing this, the moving object is a fixed distance from the camera, and the speed/acceleration of the object for each trial is nearly the same, given that gravity is the one doing all of the work. After running a few tests, it appeared that this approach produced the most consistent results, and thus we used it to conduct our experiments.

Shown below in Figure 2 is the actual motion detection program written in Python. The program accomplishes various different tasks, all of which allow us to quantify the results of our experiments that will later be used for calculations. To start things off, PiCamera motion is represented by vectors, and for our research purposes, we are concerned with the *magnitude* of said vectors. The program above calculates the magnitude of the vectors for every instance of motion detected and stores them in the variable "a". To determine when *significant* motion is detected, we first want to check if the magnitude of the present vector ("a") surpasses a certain threshold, which is 40, in this case. Then, we check if the *number* of vectors that have acheived this threshold surpass a threshold of 10, and if so, then the amount is stored in a list, and we output that motion was detected. At the end of the video, we have a list containing each significant sum. The output of the program looks something like what is shown below in Figure 3.

In order to effectively compare the peformance of the motion detection software in different environments, we needed to numerically represent the detected motion for each environment. The simplest way of doing this was having

```
import picamera
import picamera.array
import numpy as np

motionList = []
class MyMotionDetector(picamera.array.PiMotionAnalysis):
    def analyse(self, a):
        a = np.sqrt(
            np.square(a['x'].astype(np.float)) +
            np.square(a['y'].astype(np.float))
            ).clip(0, 255).astype(np.uint8)
        # If there're more than 10 vectors with a magnitude greater
        # than 40, then say we've detected motion and append motionList
        if (a > 40).sum() > 10:
            motionList.append((a > 40).sum())
            print('Motion detected!')

with picamera.PiCamera() as camera:
    camera.rotation = 180
    camera.resolution = (640, 480)
    camera.framerate = 30
    camera.start_recording(
        'motion.h264', format='h264',
        motion_output = MyMotionDetector(camera)
        )
    camera.wait_recording(7)
    camera.stop_recording()
    print(motionList)
```

Fig. 2. Python code for our motion detection program

```
Motion detected!
Motion detected!
Motion detected!
Motion detected!
Motion detected!
[17, 19, 15, 17, 12]
```

Fig. 3. Sample output of our motion detection program

a single number represent the amount of detected motion for each setting. The process for obtaining this singular value was as follows: For each video, we calculated the average of the sum list that was fully populated by the end of the video. To accomplish this, we utilized the help of Microsoft Excel's automatic average calculator for columns. The average represented the amount of motion for that video. For each setting, we ran three trials to account for any extraneous factors possibly affecting the data. Thus, at the end of the experiment, each setting had three videos and three averages. Finally, we took the average of the averages for each setting, and the resulting number was the value used to represent the amount of motion for that setting. After running all trials and obtaining the value for each setting, we simply had to compare the final values for our results and conclusions.

RESULTS

Shown below in Figure 4 is our motion data recorded in the standard lit environment. To acheive standard lighting, we didn't have to manipulate the setting, we just had to record in the normal lighting of the Robotics Lab. After performing all necessary calculations, the overall motion was **16.056**.

Shown below in Figure 5 is the motion data for the low-lit environment. To acheive low-lighting, after turning off the lights we closed all but one of the shades in the Robotics Lab to allow just enough light to pour into the room so the camera can actually pick up the motion. The overall motion for low-lighting was **13.278**.

Standard Lighting		
Trial 1	**Trial 2**	**Trial 3**
17	17	16
19	18	13
15	16	16
17	16	18
12	15	19
	15	14
16	**16.16667**	**16**

Fig. 4. Table containing motion data for the standard lit environment

Low Lighting		
Trial 1	**Trial 2**	**Trial 3**
11	11	11
14	14	15
13		18
12.66667	**12.5**	**14.66667**

Fig. 5. Table containing motion data for the low-lit environment

Finally, shown below in Figure 6 is the motion data for the environment with the dynamic background. To acheive this, we placed a monitor behind the rolling object playing a movie trailer. This would simulate the possibility of some using PiCamera in their home for surveillance, and the TV was on when a burglary took place. Additionally, a row of LED lights was behind the moving object flashing every second. All in all, we simulated "motion" behind the primary moving object. The overall motion for an environment with a dynamic background was **15.657**.

Standard Lighting w/ Dynamic Background			
Trial 1	**Trial 2**	**Trial 3**	
22	17	22	
14	16	13	
16	12	18	
15	16	15	
11	18	12	
17	13	13	
22	16	16	
12		18	
12			
15.66667	**15.42857**	**15.875**	

Fig. 6. Table containing motion data for the dynamic-background environment

Below in Figure 7 is a bar graph visually demonstrating how well the camera peformed in the different environments. Based on the results, it appears that the camera detected less motion in the low-lit environment, despite the motion itself not changing. For the dynamic background, the overall motion was only slightly less than the standard lighting, a quite negligible difference. All in all, it appears that only in the low-lit environment did PiCamera peform worse.

FUTURE WORK

Future work on this project may entail further experimentation with altering the thresholds and testing in environments not seen in this experiment. Further research could investigate expanding the amount environments tested. While

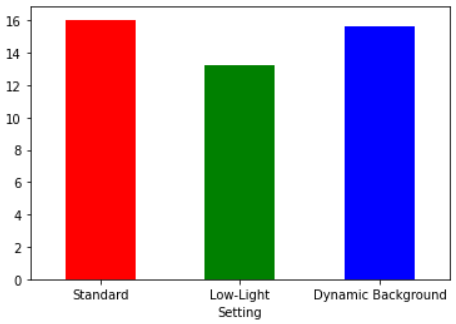

Fig. 7. Bar graph showing our results

our research only tests the program's effectiveness in the three environments described, one could choose to test the program in different light levels and some combination of the environments previously used. For example, a dynamic background in a low-light environment. Of course, in addition to changing the environment that the camera is observing, one could also consider other factors such as what is being recorded and at what distance. One question that could be asked is how far away can motion be before it is no longer detected. To truly make an effective motion detection system for something as reliant on effectiveness as a home security system, all these factors should be taken into consideration and could each be a basis for research.

CONCLUSION

Based on the data we collected from our testing, we believe that the effectiveness of the picamera motion detection can change with differing environments. If the camera is set in a low-lit environment, then it is unable to detect as much as motion as when it is recording in a well-lit environment. If one were to use picamera motion detection software in, let us say, their home for detecting potential burglaries, then they would have to ensure the software can adapt to a darker environment. While a dynamic background had little to no effect on the motion detected, this result may change depending on the values set for the thresholds. Overall, we can conclude a darker environment can alter the amount of motion detected.

REFERENCES

[1] https://picamera.readthedocs.io/en/release-1.13/recipes2.html
[2] https://www.hackster.io/ardy345678903456789/blind-hat-helper-9e4667