Project Report: Fall Detection System

[redacted]

CSCE 462-500: Microcomputer Systems | Fall 2021 | Dr. Jyh Liu

Section 1: Summary

This document details the completion of our project for a fall detection system for our CSCE 462: Microcomputer Systems course under the supervision of Dr. Jyh-Charn Liu at Texas A&M University. Using the concepts learned in this course and the skills we gained from our previous classes, we have created an efficient and cost-effective system capable of automatic and accurate detection of fall. Concepts used include knowledge of hardware components, skills in designed algorithms, and network communication. These concepts and how they are employed in our final project will be detailed throughout this report.

This project mainly utilizes the Raspberry Pi microcomputer ecosystem and all the hardware and software to which it can connect. Through the use of a breadboard, a webcam, and an inertial measurement unit (IMU, a device capable of measuring movement through gyroscopes and accelerometers), we are able to use two Raspberry Pis to create a portable system that can detect falling incidents. At a high-level, the three software subsystems in use are: color detection, motion analysis through acceleration patterns, and network communication through UDP protocol. These subsystems work alongside hardware such as a webcam, IMU, and router respectively.

The Raspberry Pi connected to the subject will hereafter be referred to in this report as the IMU RPi while the Raspberry Pi connected to the webcam will hereafter be referred to as the Color Detection RPi. The host used for the UDP protocol is a router and is the proper network that is crucial for this system to work as there are no inherent, wireless ways for these two Raspberry Pis to communicate with each other. With our project, we hope to contribute to society by offering a low-cost solution to an incredibly common problem that is present all over the world and throughout generations.

Section 2: Introduction

According to the Centers for Disease Control and Prevention (CDC), one out of every four elderly fall each year with one out of every five incidents resulting in severe injury. Many elderly people become incapable of movement, or sometimes even become severely injured, due to falling in their private residence without any immediate assistance available. Currently, there are products on the market that try to resolve this issue such as LifeAlert, a portable medical alert system. However, these types of services require monthly subscriptions and locked-in contracts. There may also be times when the elder falls unconscious or is otherwise unable to move their body. Automated and accurate fall detection is still a necessary tool for protecting the elderly.

Our project focuses on providing an affordable solution for caretakers while also allowing the elderly autonomy and independence. Our solution provides caretakers with peace of mind and the elderly with the privacy they deserve. Through the use of computer vision, movement measuring, and local network communication, we have created a system capable of detecting fall in a minimally invasive manner.

Section 3: System Design

3.1 High-Level Overview

As a high-level introduction, the fall detection system utilizes 3 key components in order to accurately detect falls: color detection, acceleration pattern recognition, and UDP transmission. These 3 components are vital to making the system work continuously however, each component can be improved which will be discussed later.

The main driver of the entire system is the color detection system, the system continuously searches for the color red underneath an optimized height threshold which was set within the Python code. Another main driver of the entire system is the acceleration pattern recognition system, the aforementioned system searches for a unique pattern which consistency appears when a fall occurs. This unique pattern will be described in more detail in the following sections.

These two systems are interconnected utilizing a UDP communication protocol, a simple but effective transmission which allows us to send a simple message from the acceleration pattern recognition system to the color detection system. Upon detecting fall, the acceleration pattern recognition system sends a message to the color detection system which creates a file within the color detection system’s operating system. The color detection system is constantly looking for this file while searching for red pixels in the still images the system captures. At the exact moment the color detection system detects both the file and a specific amount of red pixels, a buzzer sounds for 30 seconds which alerts that a fall has occurred.

In this high-level introduction, elementary verbiage was utilized in order to explain the system in a very simple manner; however, more explanation of both the software and the hardware will be provided in the following sections.

3.2 Hardware

In the fall detection system, there are multiple key pieces of hardware that are being utilized in order to detect fall: 2 Raspberry Pis, Adafruit MPU-6050 (IMU), Hrayzan 1080p Webcam, 10,000 mAh power bank, TP-Link router, and a buzzer. The color detection system which was stated in the previous section utilizes the Hrayzan 1080p Webcam, buzzer, and one of the Raspberry Pis. This system is also encased in a small box which holds only the Raspberry Pi while both the buzzer and the webcam are situated externally.

The acceleration pattern recognition system which was stated in the previous section utilizes the Adafruit MPU-6050 (IMU), 10,000 mAh power bank, and one of the Raspberry Pis. This system is encased in a 3D printed capsule which will be tightly strapped onto the body of the person for which fall detection is being conducted. The UDP transmission requires the TP-Link router in order to send a simple message across a mutual intranet which both Raspberry Pi's are connected to. The color detection system requires the camera since it is taking still images of the nearby environment at around 1 frames per second.

The system also requires the buzzer since this system will sound the buzzer once both of the systems have reported a fall. The acceleration pattern recognition system requires the Adafruit MPU-6050, an IMU, in order to accurately detect XYZ acceleration; this data is critical to the system in order to detect falls. Furthermore, this system will also be utilizing the portable power bank in order to power both the IMU and the Raspberry Pi while allowing for full mobility throughout the environment.

3.3 Software

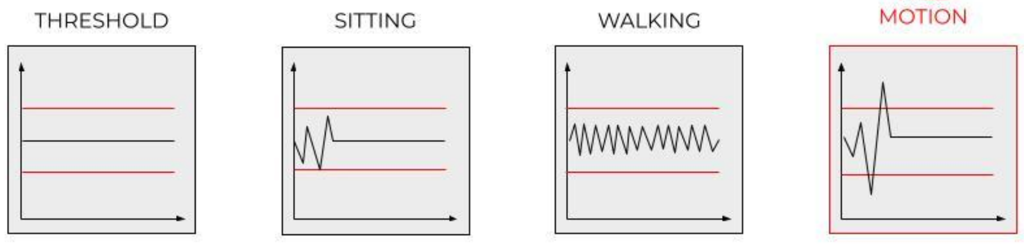
3.3.1 Accelerometer

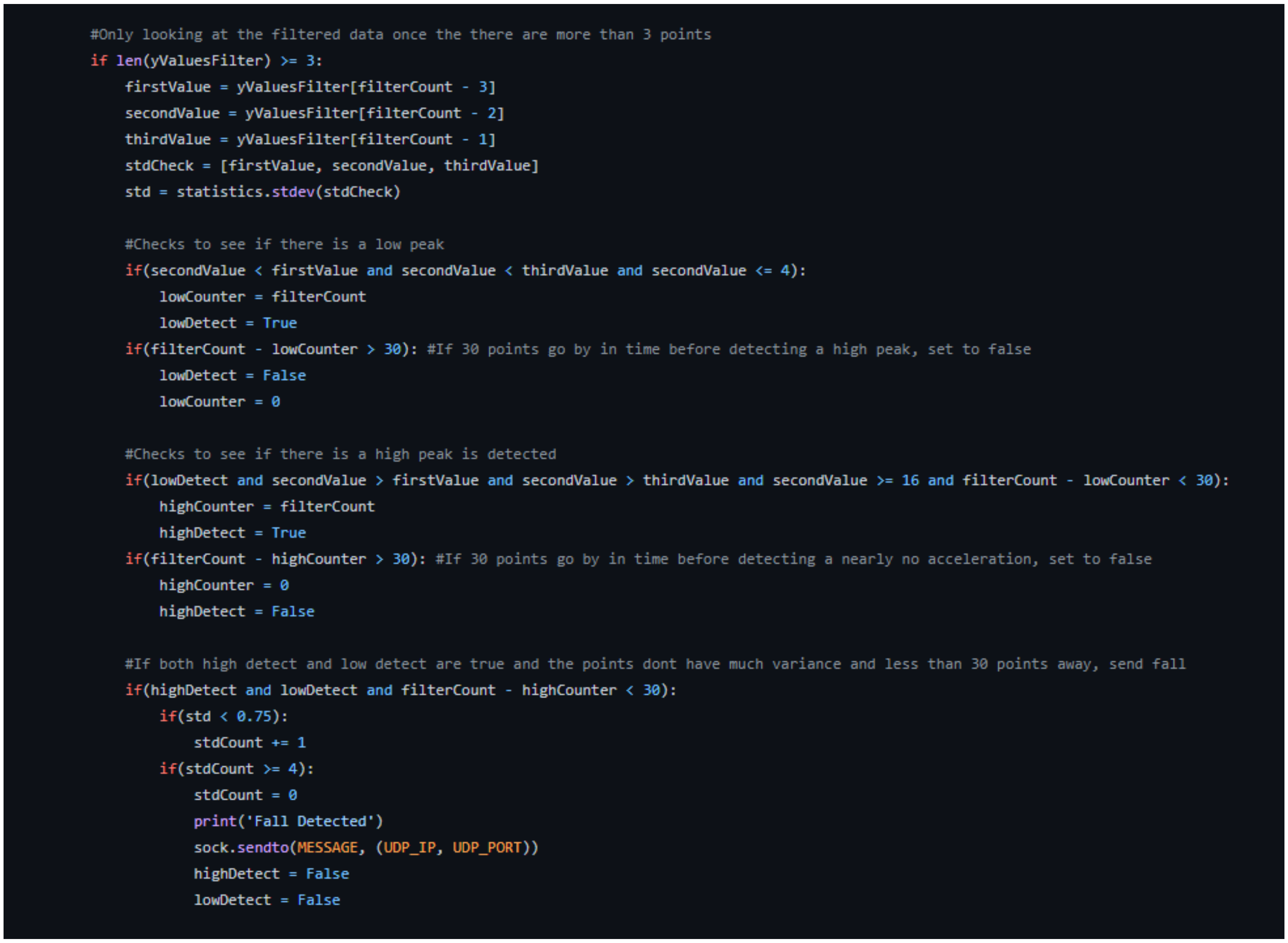
The software which controls and drives the accelerometer is seen in "Accelerometer.py", this file can be found in the google drive in a folder called "Source Code". As a brief overview of the Adafruit MPU-6050, the device provides accurate acceleration data in the X, Y, and Z directions which the acceleration pattern recognition system analyzes in order to detect the aforementioned unique pattern. The unique pattern, which has been briefly discussed, appears nearly every time a fall occurs; this has been proven through multiple experiments. Furthermore, this unique pattern consists of 3 key sequential points: a very low peak, a high peak, and a multitude of points with a standard deviation of under 0.75. These 3 key points allow us to consistently and accurately detect a fall; however, these key points also appear in another form of movement.

The movement in question is the action of sitting however, the low peak which appears in the action of sitting never dips below 5 units which allows us to differentiate betweens the action of falling and sitting. In more detail, our algorithm first filters all incoming data, it takes the mean of every 3 data points and adds it to a new array of filtered data points. Next, the algorithm ensures that there are more than 3 points in the array of filtered data points in order to begin the analysis step.

The analysis step is a critical series of if statements which allow us to detect falls. The first step is detecting the low peak which must be followed closely by a high peak. If a high peak is not detected within 30 points, the analysis stage will be reset. If a high peak is detected, a narrow range of values with extremely low variance must be detected within 30 points, if these values are not detected, the analysis stage will be reset. If this narrow range of values are detected, the code will enter the final step which includes sending a simple message to the color detection system utilizing a UDP communication protocol. The message is transmitted nearly instantaneously and at this point the main control of the system has passed over to the color detection system.

The image below shows the difference between common patterns which the IMU could potentially see and how it must detect falling and falling alone. Also a small snippet of code is provided below which shows the series of if statements which were mentioned previously.

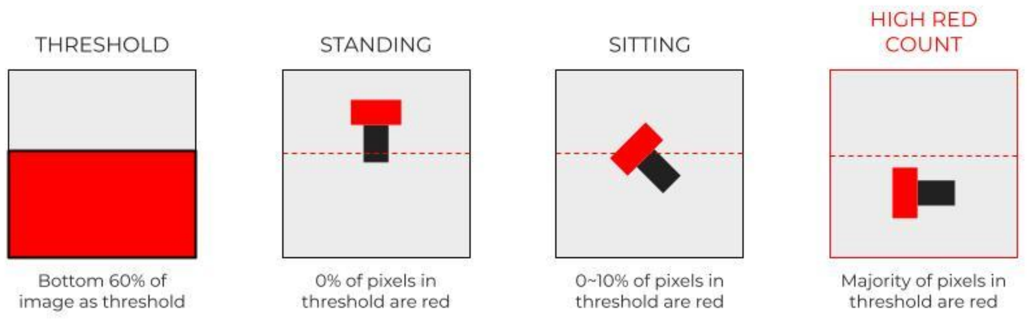




```python
#Only looking at the filtered data once the there are more than 3 points
if len(yValuesFilter) >= 3:
    firstValue = yValuesFilter[filterCount - 3]
    secondValue = yValuesFilter[filterCount - 2]
    thirdValue = yValuesFilter[filterCount - 1]
    stdCheck = [firstValue, secondValue, thirdValue]
    std = statistics.stdev(stdCheck)

    #Checks to see if there is a low peak
    if(secondValue < firstValue and secondValue < thirdValue and secondValue <= 4):
        lowCounter = filterCount
        lowDetect = True
    if(filterCount - lowCounter > 30): #If 30 points go by in time before detecting a high peak, set to false
        lowDetect = False
        lowCounter = 0

    #Checks to see if there is a high peak is detected
    if(lowDetect and secondValue > firstValue and secondValue > thirdValue and secondValue >= 16 and filterCount - lowCounter < 30):
        highCounter = filterCount
        highDetect = True
    if(filterCount - highCounter > 30): #If 30 points go by in time before detecting a nearly no acceleration, set to false
        highCounter = 0
        highDetect = False

    #If both high detect and low detect are true and the points dont have much variance and less than 30 points away, send fall
    if(highDetect and lowDetect and filterCount - highCounter < 30):
        if(std < 0.75):
            stdCount += 1
        if(stdCount >= 4):
            stdCount = 0
            print('Fall Detected')
            sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
            highDetect = False
            lowDetect = False
```

3.3.2 Color Detection

The software which controls and drives the Hrayzan webcam is seen in "ColorDetector.py", this file can be found in the google drive in a folder called "Source Code". The npm package which allows us to control the camera is called "fswebcam", it allows us to take still images of the nearby environment at 1 FPS. Our main method of detecting a human figure is through the color red, it is assumed that the person for which detection is being conducted will be wearing a red article of clothing on the upper half of their body such as a jacket or t-shirt. Furthermore, in the beginning of the algorithm, a brightness filter is applied in order to better detect the color red and a height threshold is applied which essentially crops the image and only analyzes the cropped image. The reason for the height threshold is in order to detect the color red closer to the floor and avoid seeing the color whenever the person in question is standing or sitting. After the brightness filter and image cropping is complete, the algorithm proceeds to find all red colors which lie within a red color boundary that we have set. More specifically, we look for all colors and increase a counter when the color in question is between rgb(125, 50, 50) and rgb(255, 50, 50). The reason for this color boundary is because all colors will not be the maximum saturated version of red, they all have certain amounts of green and blue which must be accounted for. Furthermore, this color boundary was optimized throughout extensive testing in order to capture the most amount of red. In totality, this counter is used in parallel with an internal boolean variable which states if the packet from the acceleration pattern recognition system has been received. If both the counter is greater than 50 and the packet exists, the buzzer will sound for 30 seconds until the packet is deleted and another fall can be detected. The image below shows how the color detection algorithm works and is able to detect a majority of the red pixels below the height threshold.

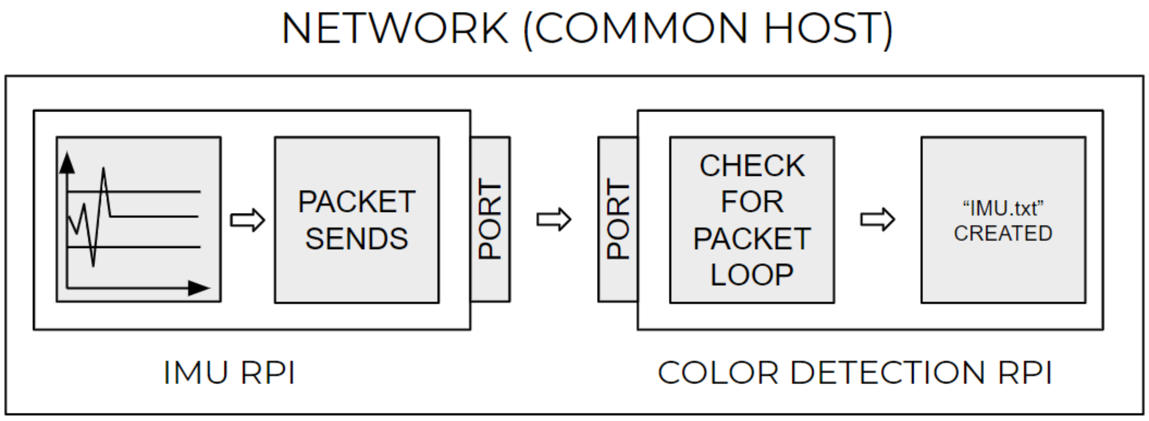

THRESHOLD
Bottom 60% of image as threshold
STANDING
0% of pixels in threshold are red
SITTING
0~10% of pixels in threshold are red
HIGH RED COUNT
Majority of pixels in threshold are red

3.3.3 UDP Communication

The UDP communication protocol allows us to send a packet of data from the acceleration pattern recognition system to the color detection system. In more detail, the color detection system is the “receiving” end of the UDP transmission while the acceleration pattern recognition system is the “sending” end of the UDP transmission.

Before continuing, it is important to note that the system can only be interconnected through the use of the TP-Link router, the router acts as a bridge between the two Raspberry Pi’s allowing for unidirectional packet transmission. Furthermore, the color detection system binds its IP address and a chosen port together which is exposed by the router that allows it to receive incoming traffic. This means that any machine, connected on the same network, can send the color detection system packets of data as long as the machine has the color detection system's IP address and port number. The previous statement essentially explains how the acceleration pattern recognition system sends the color detection system information. The acceleration system uses the color detection system's IP and port and sends one simple encoded message, "IMU".

On the receiving end, the color detection system constantly listens to incoming traffic, if at any given time it receives, "IMU", it creates a file in the same location as the "ColorDetector.py" file. The aforementioned python code is then able to understand that the IMU has detected the falling motion pattern. The image below shows a visual demonstration of the IMU RPI connecting and sending data to the color detection RPI which in turn creates a file called "IMU.txt".



Section 4: Application and Future Improvements

The purpose of developing an automated fall detection system is to relieve people from the fear of falling and not receiving the proper care or attention on time. Leveraging the power of utilizing microcomputers and sensors and the ideas from computer vision, our team was able to develop a fully working prototype for a fall detection system that would be useful for many. Every year about 20000+ workers from multiple industries are injured and nearly 100s die from the lack of immediate attention. Elderly individuals have similar statistics, where they are severely injured with no immediate care. We presume that this would be mostly utilized by elders as it takes away their ongoing fear of not getting the help they need when fallen, given that they are alone or incapable of calling for help on their own. Developing a fall detection prototype is one of the first steps to create a comprehensive safety tool for people, even in the most extreme cases.

Throughout our developmental stage there are a number of improvements or additions that could be made for efficiency and convenience of the system. If there were to be a second phase to this prototype, the system could possibly detect more than just red using a similar algorithm to what we have created in this project. Another option is to create a LED detection system. For example, GREEN could mean there's no fall detection or the environment is clear, RED could be that there's a falling motion sensed, and flashing lights could mean that there is an emergency or fall detected and that's when the buzzer goes off.

An improvement that should be prioritized is a button system to delete the IMU.txt file and stop the buzzer. Currently, we just have programmed the buzzer to automatically stop after 30 seconds and the IMU.txt file to be deleted, however a more useful approach would be to implement a button, where when pressed it will stop the buzzer knowing that there is someone there to help. This is more effective because it can take more than 30 seconds for someone to be notified and get proper help, but if the buzzer automatically turns off, there is a chance that the situation is not taken care of.

Lastly, an extension to this project, for market availability, would be sending text messages to notify that someone has fallen. There could also be an app, where the user could store multiple reliable friends' or family members' numbers, so in case of an emergency those people would receive a text message notifying that a fall has been detected and requires their attention.

Section 5: Conclusion

At a high-level, the fall detection system is divided into three subsystems which include recognizing acceleration patterns, color detection and network communication. These software systems are supported with hardware components, such as the IMU to measure the movement, a webcam to detect the color red, a router for UDP transmission, and two Raspberry Pi's that make up for our environments to run and test the fall detection system. Using these components, we were able to successfully integrate these three subsystems using a router as the main host.

First we were able to detect different acceleration patterns and filter out the data, so it can accurately detect a fall as compared to other movements. When fall is detected, we were able to send a text file to the main Raspberry Pi using UDP protocol as a form of communication between the two microcomputers. Finally we implemented a color detection algorithm with the use of a webcam, where it would take an image of our environment, crop it for some height threshold and iterate through pixels specifically looking for the color red. To finally bring these together, we checked that if a file called IMU.txt was sent and the color red was picked up then the person had fallen.

Integrating these subsystems increased our accuracy to nearly 100%. There are occasional false positives, but there's a list of improvements discussed in section 5 that could perfect this fall detection system given there were to be a phase 2 where we had more time and resources to experiment.

Appendix 1: Important Resources

1. Youtube video which helped us greatly by explaining exactly how to connect the two Raspberry Pis by using the UDP communication protocol and a similar host.
   [redacted]
2. This is the Github repository for all of the source code, it holds the source code for the accelerometer, the color detection system, and the receiving end of the UDP connection.
   [redacted]
3. This is the Google Drive link where all of our final project materials are stored. It includes the final report and presentation, source code, and the video clips.
   [redacted]
4. This link helped us understand how to use the webcam since we were unable to use the camera within the python code itself, it gave us another option which included command line.
   Link: Using USB webcams | The Raspberry Pi Guide (raspberrypi-guide.github.io)

Appendix 2: Slack

[redacted]

[redacted]

[redacted]

[redacted]