Smartphone Application to Detect Texting While Driving

Final Project Report

Group:
sdmay19-16
“The Face Mates”

Client:
Daji Qiao

Faculty Advisor:
Daji Qiao

Team Members:
Ryan Baker - Lead Architect
Derek Clayton - Report Manager
Lucas Golinghorst - Test Engineer
Andrew Knaack - Lead Designer
Sara Mace - Meeting Scribe
Kristina Robinson - Project Lead

Team Website:
sdmay19-16.sd.ece.iastate.edu

Submission Date:
4/29/2019

TABLE OF CONTENTS

List of Tables

List of Figures

List of Definitions

- False positive: In this case, a false positive is our application incorrectly identifying a passenger as a driver and blocking their texting capabilities.
- False negative: In this case, a false positive is our application incorrectly identifying a driver as a passenger and not blocking their texting capabilities.
- TwD: An acronym which stands for Texting while Driving

0. Executive Summary

This project was designed as a research project to address the issue of texting while driving. According to teensafe.com, just 3 years ago, 391,000 injuries were caused by distracted driving accidents. Even more serious than that, 9 people are killed everyday from distracted driving accidents [1]. Answering texts while driving might seem harmless, but replying to a text message takes the driver's eyes off of the road for at least a few seconds. It takes only three seconds of the driver having their eyes off of the road to be in a car crash. While texting and driving is not the only cause of distracted driving, approximately 660,000 drivers use a cell phone during the daytime [1]. That is a significant amount of people texting while on the road which will make it more likely for them to get in a car crash. In order to find a solution to this problem, our group will build an android application to detect if someone is texting while driving. Accurately detecting whether someone is in the driver's seat and that they are texting while driving is our biggest challenge. In order to do that without simply locking out everyone from their phones, our groups solution will have to include many different measures to ensure accurate detection.

The solution to this problem will be to develop an android application to detect whether someone is TwD. Once TwD has been detected, the user will not be able to send text messages until they stop driving. We will ensure that we can accurately detect TwD by checking multiple different categories. The first thing we will check is if the driving is moving faster than 10 miles per hour. This will tell us if they are in a moving vehicle, regardless of direction (including reverse). The second category the app will look at is determining where the person is sitting in the car. The application will use centripetal acceleration to determine if a user of the application is sitting on the left or right side of the vehicle.. Our last category of verification is learning how the user normally uses their phone so the app will be able to find differences when a user is TwD.

To keep the scope manageable, this project has been designed to work with a proprietary texting application that our group created.

1. Requirements Specification

1.1 Functional Requirements

1. The application prevents drivers from sending texts if certain dangerous conditions are met. This is the most important functionality and the main purpose of the project.
2. The false positive rate and false negative rate should not be greater than 10%. It would be very easy to block 100% of drivers, but that would also block passengers as well. As many bad drivers should be blocked as possible without also accidentally preventing texting passengers from driving.
3. The application should run on Android OS 6.0 and newer. As many phones as possible should be able to run the app without being limited to an unnecessarily basic version of Android. The application should be available to as many as possible, and with Android, older versions of software are common.
4. The app does not require an Internet connection (it is a standalone app). This is necessary if service is poor and to avoid breaking any applicable data handling laws with outside data storage. Using internet connection only improves the accuracy of calculating vehicle speed, but is not mandatory for the application to work.

1.2 Use-Cases

1. The user is driving at speeds above 10 miles per hour and trying to type a text message. The application prevents the user from continuing their message by wiping out their input.
2. The user is in a car at speeds above 10 miles per hour but is not the driver and is using their phone to text. The application should not prevent this user from texting.
3. The user is walking and using their phone to text. The application should not prevent this user from texting.
4. The user is not moving and using their phone to text. The application should not prevent this user from texting.

1.3 Non-Functional Requirements

1. The app itself will be easy to set up and run. The app should be as easy as possible for the user to set up. This does not count hardware, as hardware will probably be a tedious setup and can't be helped.

2. The app must be reliable at least 95% of the time, meaning that it runs consistently without crashing. This is important since the app is active most hours of the day to monitor user habits; also, obviously it would defeat the app’s purpose entirely if it crashed while the user was driving.
3. The app must comply with applicable sensitive data laws (presumably resolved by making it a standalone app). Personal information is potentially at risk, so it is important that our app does not break the law when dealing with that sensitive data.

2. System Design & Development

2.1 Design plan

The design was intended to incorporate six modules to determine one thing: is the phone user texting while driving. The six modules planned initially were: Texting Speed, Spell-Checking, Phone Handling, Centripetal Acceleration, and Camera. Texting Speed and Spell Checking modules would determine if the user is behaving abnormally while texting. The Centripetal Acceleration and Camera modules would determine if the user is in the driver's seat. The combination of possible danger, distraction, and the user being in the driver's seat results in locking the phone's texting abilities for safety reasons. As the project proceeded, some modules were eliminated due to infeasibility: Phone Handling and Camera. Therefore, the final design plan includes only four modules, elaborated below:

Speedometer: The first and most fundamental is the speedometer, a sensor Android phones have built-in. This will determine if the system is trying to detect texting while driving at all. If the speedometer is below a safe speed, the system will switch to its learning mode. If it is above safe speed, the system will switch into detection mode and begin calculating the probability of texting while driving. The safe speed for the application is defined as at or below ten miles per hour. Its inputs are readings from the phone's built-in linear acceleration sensors. Its output is a true/false value indicating if the user is traveling at a potentially dangerous speed.

Texting Speed: When not in detection mode, key input will be captured while the texting application is in use so that the system can learn the user's typical texting speed. During detection mode, the system will take this norm and compare it to how fast the user is currently typing. If the texting speed is significantly slower, it implies the user may be distracted (possibly driving). Its first input is the danger indication from the Speedometer. Its second input is the current message the user is typing. If the danger value is false (no danger), then its output is a recalculation of the user's typical average texting speed with the addition of the analytics of the current message. If the danger value is true (possible danger), then its output is a true/false indication of distracted driving based on a comparison between the current texting speed and the recorded average.

Spell-Checking: When not in detection mode, the user's captured key inputs will be analyzed for spelling errors so that the system learns how many errors are in the average sentence the user types. During detection mode, the system will take this norm and compare to the number of errors in what the user is currently typing. If the ratio of misspelled words is significantly higher, it implies the user may be distracted (possibly driving). Its first input is the danger indication from the Speedometer. Its second input is the current message the user is typing. If the danger value is false, then its output is a recalculation of the user's typical spelling correctness percentage with the addition of the analytics of the current message. If the danger value is true, then its output is a true/false indication of distracted driving based on a comparison between the most recent spelling correctness analysis and the recorded average.

Centripetal Acceleration: Using the gyroscope and magnetometer, it can be determined what side of the vehicle the user is on when it makes a turn. The gyroscope is used to determine when a vehicle is turning while the magnetometer is used to determine what side of the car the user is on. If the sensor readings show that the user s on the right side of the vehicle, then the user cannot be in the driver's seat and detection mode will be disabled. If module determines the user is in the back, then detection mode will also be disabled. If the user is determined to likely be the driver, spelling and texting speed indications are much more likely to disable the phone's texting functions. Its first input is the danger value provided by the Speedometer. Its second input is the data provided by the gyroscope. Its third input is the data provided by the phone's magnetometer sensors. If the danger input is false, this module is dormant and produces no outputs. If true, then its output is a true/false value indicating if the user is in the right (true) or left (false) side of the car.

2.2 Objectives, Constraints, & Tradeoffs

2.2.1 Design Objectives

The objective of our project is to make a reliable mobile application for detecting and preventing texting while driving. To accomplish this, we will create an application for Android phones which will use built-in sensors and learned user texting habits to reliably know when to lock texting functionality on the phone. The application will be a standalone application, thus not requiring any additional hardware. The end product will have a low false positive and false negative rate so drivers cannot accidentally be allowed texting anyway, and passengers will still have full functionality of their phones.

The primary goal of this solution is to detect when a driver is texting and driving. The secondary goal is to implement a lock out process that is triggered when texting and driving is detected. This in turn will create a safer driving experience for the user and the other drivers on the road.

2.2.2 System Constraints

1. The application must be compatible with Android 6.0 and onwards. This will allow more phones to be able to use our application without compromising functionality.

2. Some useful sensors (such as the Accelerometer) that are built into Android hardware cannot be used reliably because they require calibration for every device.

2.2.3 Design Tradeoffs

Design Strengths:

- Modular build allows the four modules to cover each other's weaknesses and inaccuracies.
- Modular build allows for easy removal of parts of the application deemed infeasible later.
- Some modules can fail and the application will still be functional.
- No extra hardware infrastructure is required for any of the modules to function properly.

Design Weaknesses:

- Modular nature requires work on many parts, which will be hard to implement in the limited window of time given.
- Modules require specific hardware components and are rendered completely useless if the component they require is damaged.
- Design relies on phone sensors which have varying sensitivities across different phones.

2.3 Architectural Diagram

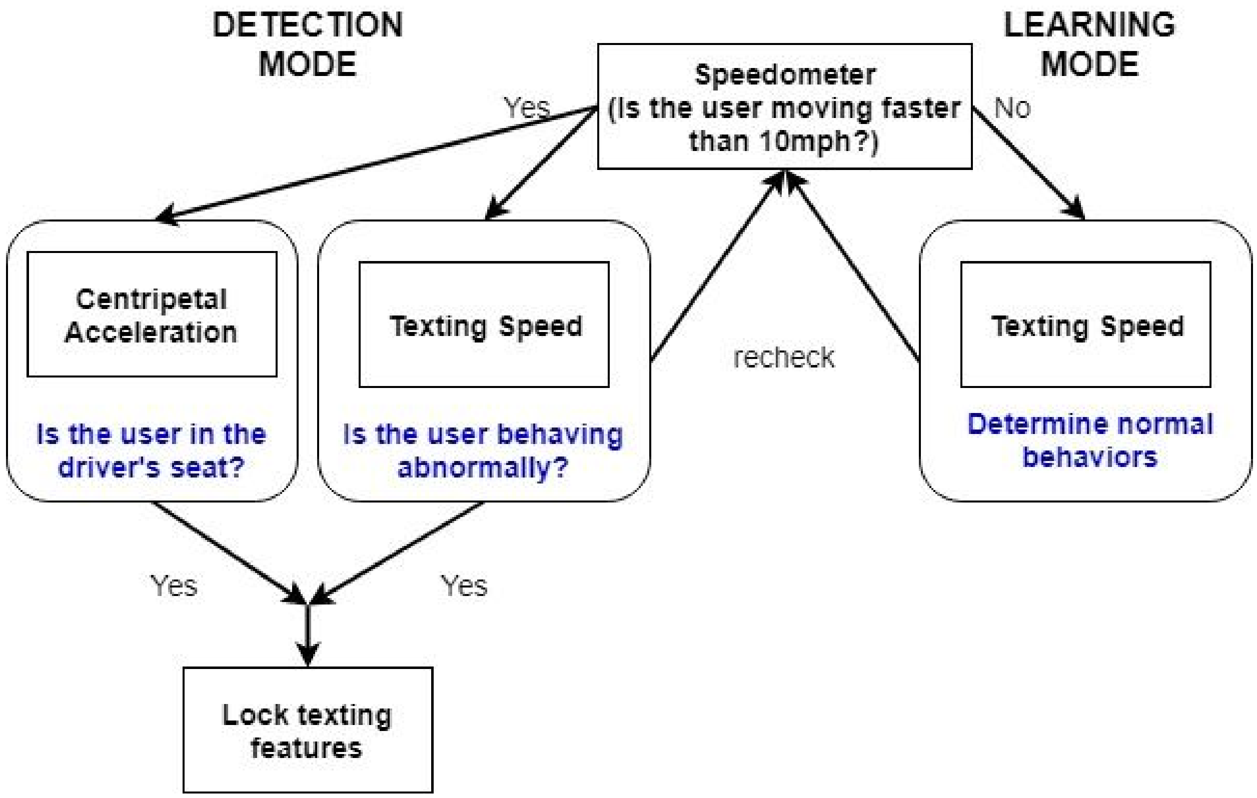
DETECTION MODE
LEARNING MODE
Speedometer (Is the user moving faster than 10mph?)
Yes
No
Centripetal Acceleration
Is the user in the driver's seat?
Texting Speed
Is the user behaving abnormally?
recheck
Texting Speed
Determine normal behaviors
Yes
Yes
Lock texting features

Figure 1: Architectural Diagram

The diagram above is the architectural view of the modules. It should make clear how the modules relate to each other. Notice the only module running during Learning Mode is the Texting Speed. Speedometer is always being rechecked for speed updates.

2.4 Description of Modules, Constraints, and Interfaces

2.4.1 Modules

The project originally consisted of 6 modules, only 3 of which could be successfully implemented in the final design.

Speedometer Module: A successful module. This module uses Location Services to calculate the user's movement speed. If the user is going faster than 10 miles per hour, this is considered a potentially-unsafe circumstance. The behavior of other modules changes based on whether the user's speed is below or above this threshold.

Texting Speed Module: A successful module. This modules uses a Text Watcher to monitor the text box into which a user types text messages. If in Learning Mode, this module records typical user texting speeds in local data and calculates an average. If in Detection Mode, this module compares the user's current speed to the expected average. If the current speed is below 70% of the expected average, this is considered a potentially-unsafe circumstance as it indicates the user may be driving (and therefore distracted, causing the slower typing).

Centripetal Acceleration Module: A successful module. It works off the fact that users on the driver and passenger side experience different levels of centripetal acceleration. Using two sensors, the gyroscope and magnetometer, it first detects if the user is turning and then what side of the vehicle the user is on. It detects left or right turns with the gyroscope sensor and it detects driver/passenger side using the magnetometer readings.

Spell Checker Module. An unsuccessful module. This module was intended to use the Spell Checker API of Android and the Text Watcher already seen in the Texting Speed Module to create an expected average of spelling correctness similar to how that module tracks average speed. It was infeasible to design due to the fact that Android phones no longer support the required library, and no viable solution has replaced it.

Phone Handling Module: An unsuccessful module. This module used a built-in magnet sensor to determine changes in how the user is handling their phone. Like the Texting Speed Module, it would record typical holding angles in Learning Mode and detect abnormalities in Detection Mode. However, the readings of the magnet sensor proved to be too insignificant to reliably tell how the phone orientation was changing.

Camera Module: An unsuccessful module. This module would have used the phone's front-facing camera (towards the driver) and machine-learning techniques to identify which seat the user was sitting in. A much more significant amount of time was put into this module compared to the other two unsuccessful modules. However, the technology was not suitable for identifying one specific object to the exclusion of any others, and many thousands of images of car interiors would have been required for accurate results.

Three simultaneous danger readings from the Speedometer, Texting Speed, and Centripetal Acceleration modules indicate that the user is texting while driving, causing the Texting Speed Module to prevent the users from further texting by forcefully erasing their text inputs.

2.4.2 Constraints

1. Actual texting while driving cannot (or at least should not) be performed for testing purposes because it is dangerous behavior, so there may be some variation in the data between testing and when trying to approximate when someone is *actually* texting while driving.
2. Testing the app most accurately requires at least two people with an actual car driving around. This is primarily a time constraint, since this type of testing is more time-consuming than testing that can be done from a computer, and both participants must be available.
3. The project is intended for research, not marketability; therefore, a fully functioning product may not result.

2.4.3 Interfaces

Pre-existing interfaces for the Android API were used for some modules. The Text Watcher API was used for the Texting Speed Module because it provided an easy way to detect when the user was typing and alter that text (namely, blanking out the textbox while driving). Location Services is another pre-existing interface used to provide information for the Speedometer Module. Its built-in speed getter is unreliable, but directly comparing the distance and time between two location calculations was sufficient to provide a usable speed.

3. Implementation

3.1 Implementation Diagram, Technologies, & Software

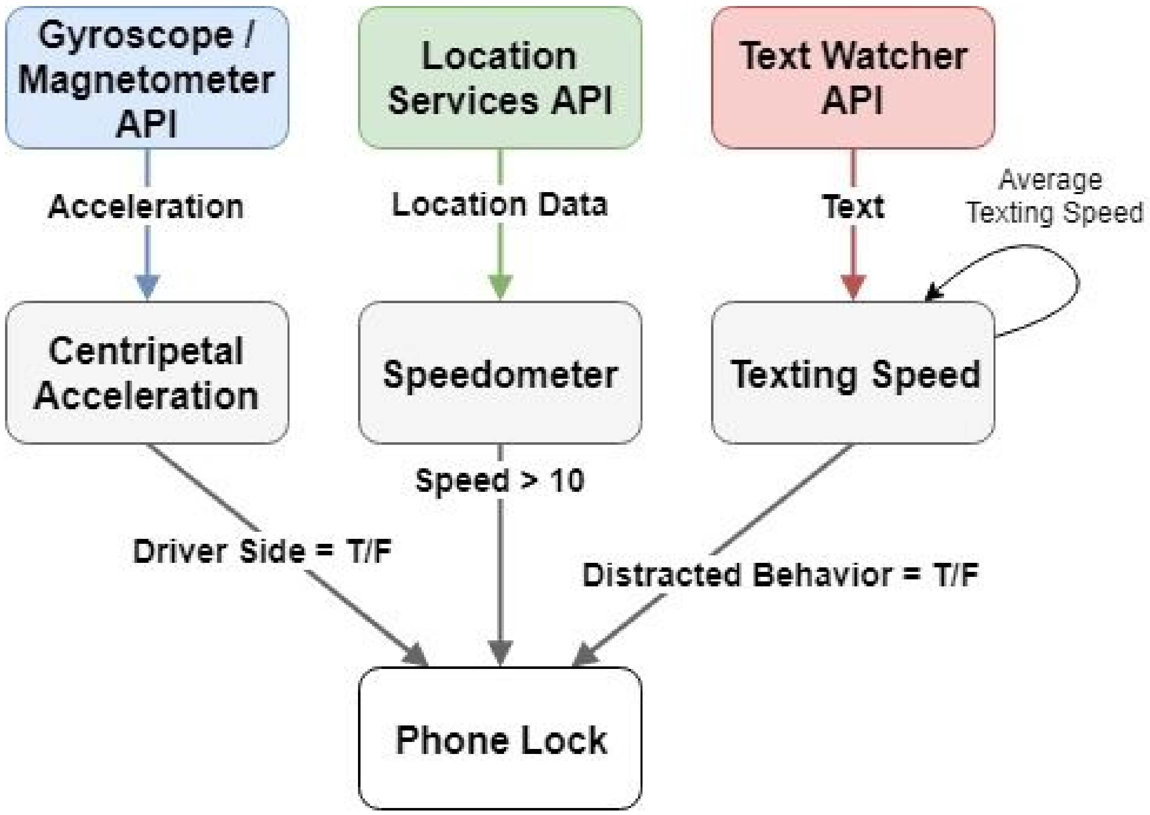

Figure 2: Inputs/Outputs & Implementation Diagram

The above diagram shows all the inputs and outputs from both the modules and the APIs which assist them. As mentioned above, three of the modules did not make it into the final design. Note that the Speedometer does not feed directly into Texting Speed or Centripetal Acceleration. The node "Phone Lock" represents the main activity of the application which holds all three modules. It gets all the information from the modules, determines what they should be doing, and how to act based on their results. The technologies and softwares that we used for this project was the actual Android phones themselves, Android Studios, Java, Gyroscope/Magnetometer API, Location Services API, and the Text Watcher API.

3.2 Rationale for Technology & Software Choices

The rationale for all the above technologies was either of necessities or convenience. We used Android phones because that is what we had available to us for testing. For that reason we had to use Android Studios to program an application for the phones. We used Java because that is the language used in Android Studios. We also used the Gyroscope/Magnetometer API, Location Services API, and the Text Watcher API because they were already built into Android Studios and easy to use.

3.3 Applicable Standards and Best Practices

One standard that will almost certainly be used is the IEEE standard for Software Unit Testing. We plan to define specific unit tests and the minimum code coverage for these tests, as well as the tools/modules that we will use to unit test code with.

Another standard that we may or may not use is the IEEE 16085-2006 -ISO/IEC 16085:2006, Standard for software engineering- software life cycle processes -risk management. It is planned to implement where possible, but most risk management happens at the beginning of a project like this. In fact, in most cases, this was impossible to implement because many choices and decisions that would be covered by this standard were made already by the client. It is possible to implement this standard when considering adding or subtracting modules for the design, and it be used there to help determine if these actions are a good idea.

Some best practices we implemented were modular coding and module testing. Each component of our program was designed to be stand alone runnable, allowing for individual testing and troubleshooting. The result was a highly modular final product that is easy to test and modify.

4. Testing, Validation, and Evaluation

4.1 Test plan - testing method, test case selection and test coverage

Testing Method: Manual testing. Our modules require user created input to generate an output.

Test Case Selection: We selected test cases by thinking about how the application would be used by the user. Each module has applicable test cases that are explained in the Unit Testing section (4.2). The system test cases were done by using the application in the field and testing the outputs.

Test Coverage: Each of our modules was individually tested for accuracy and validated. After integration our entire system was again tested for accuracy.

4.2 Unit (Module) testing

Speedometer: The application was programmed to keep a printed log of all recorded speeds. Tests were done by having two people driving in a car and monitoring the readout, comparing the readings to the actual speed from the car speedometer. Test zones used include downtown Ames, Des Moines, and on I-35.

Texting Speed: The module was tested through using Mario Kart 8 DX for Nintendo Switch. This allowed the tester to simulate texting while driving. The slower carts and the most level, plain map was used to create a realistic driving experience. The tester attempted to text predetermined messages without crashing, and the texting speed data was collected. The data resulted in the “distraction threshold” speed of 70% of the user’s normal texting speed.

Centripetal Acceleration: The module was tested by field testing the application. The driving route consisted of making consecutive left turns, then consecutive right turns. The passenger held the phone on the driver side or the passenger side, and the driver conducted at least ten turns. This process was repeated multiple times and the output of the module was compared against what was actually happening. The application succeeded in determining left or right turns nearly 100% of the time, and successfully determined driver or passenger just over 50% of the time.

4.3 Interface testing

The application interface consists of an application that operates similar to a mobile messaging application. This interface was tested manually to make sure that it functioned properly (allowed the user to create text messages and "send" them) and that it did not crash.

4.4 System integration testing

Each module was implemented independent of the other modules and tested according to the functionality it would bring the application as a whole. The texting speed module was the first module integrated into the system since it did not require the use of a car to test. This module was tested by simulating distracted texting behavior and making sure that the application would lock the user out if their texting speed dropped below 70% of their average speed. Once this module was integrated successfully, the speedometer module was added to the system. To test the integration of the speedometer module, the team rode on the cyride busses to make sure that the application would block texting if it was going faster than ten miles per hour. Then the centripetal acceleration module was integrated into the application. The team monitored the phones recognition of driver and passenger by test driving the application. If the user was identified as a passenger, the application would not block texting. If the user was sitting on the driver side of the vehicle, the application would block the user from texting with an accuracy of 54%. Once the three modules were integrated successfully, the team was able to test the entire system.

The entire system was tested by allowing all three modules to work together to determine if the user of the phone was in a car, was on the driver's side of the car, and had different texting behavior than normal. The results from the full system testing were as follows. The application correctly identified that passenger 100% of the time so they never got blocked from being able to text. As for the driver, the application correctly identified the driver 60% of the time only allowing them to text 40% of the time. This resulted in a false positive rate of 0% and a false negative rate of 40%. These false positive and false negative rates met the team's fourth of five milestones of false positive and false negative rates at or below 50%.

4.5 User-level testing

Once all modules were integrated, we went on test drives to make sure that when the appropriate conditions were met, the application would block texting. The test drives consisted of the driver operating the car in a normal driving fashion and the passenger using the application in either the front passenger seat or the back seat on the driver side. These test cases created a scenario similar to how a real user would use the application.

4.6 Validation and Verification

Our group utilized a variety of resources to ensure that our data and testing was valid. One way we did this by using multiple different android devices to check the results of our app at each stage of testing. We also verified the results of the different tests by doing debug statements, that printed to the terminal in android studio. We also added a spot in the app when we were testing where we could print values that the different sensors were printing. We were able to verify that our application met our standards that we set because we compared results we got in testing with the results we set as goals to meet.

4.7 Evaluation

Our testing was largely successful. The testing for the Speedometer was a 100% success rate. This was done using by a team member going on Cyride and logging the results of the speeds that were calculated by using location services both with and without wifi. The centripetal acceleration module was also 100% successful in identifying whether the car was taking a left or right turn, and with that information was successful 54% of the time determining if it was a driver or passenger that was holding the phone. The testing on this was done by a passenger holding the phone on the passenger's side or holding it over where a driver a driver that is texting and driving would likely hold it, and then recording if the application could correctly identify the position of the phone for both left and right turns. This was done for both the driver and passenger. The last module, the texting speed module has been shown to be very reliable and has worked 100% of the time identifying a decrease of 70% of the normal texting speed. This was determined to be the best number to measure distraction through testing. This involved users playing Mario Kart to simulate distracted driving. After all the data was measured, it was found that when distracted, texting speed fell to around 70% of normal texting speed. The whole application put together resulted in a 0% false positive rate and a 40% false negative rate. The testing for this was accomplished by having texters placed in the back seat behind both the driver and the person in the front passenger seat. They would then text and the driver would drive over the required speed and make a few left and right turns. It was then recorded if either the user got locked out of texting or not.

5. Project and Risk Management

5.1 Task Decomposition, Roles, & Responsibilities

The project was broken up into four major components. These components are the proprietary texting application, the speedometer module, the texting speed module, and the centripetal acceleration module. The team consists of six members whose roles are broken down as follows. Kristina was the project lead, Sara was the meeting scribe, Derek was the report manager, Andrew was the design lead, Lucas was the test lead, and Ryan was the lead architect. As for the parts of the project the team members worked on, Kristina and Sara were responsible for developing the proprietary texting application and integrating all modules into the system. They also worked on the phone handling module until it was determined not viable for the project. Andrew was responsible for developing the texting speed and speedometer modules. Andrew also worked on the spell checker module until it was determined not viable for the project. Derek, Lucas, and Ryan were responsible for developing the centripetal acceleration module. Lucas and Ryan also worked on the image recognition module until it was determined not viable for the project.

5.2 Project Schedule - Gantt Chart (proposed vs. actual)

Figure 3 below was our original proposed gantt chart. This schedule did not account for overlap in developing each module and quickly became different as we went through the rest of the project. We determined that some modules were not viable for our project which allowed us to put more resources on other modules. Some modules took much longer to develop than expected so we did not have time to implement machine learning into our application. Most of the semester was spent developing the proprietary texting application and the three modules. The last few weeks we spent integrating and testing the modules, integration of all modules and testing the entire system.

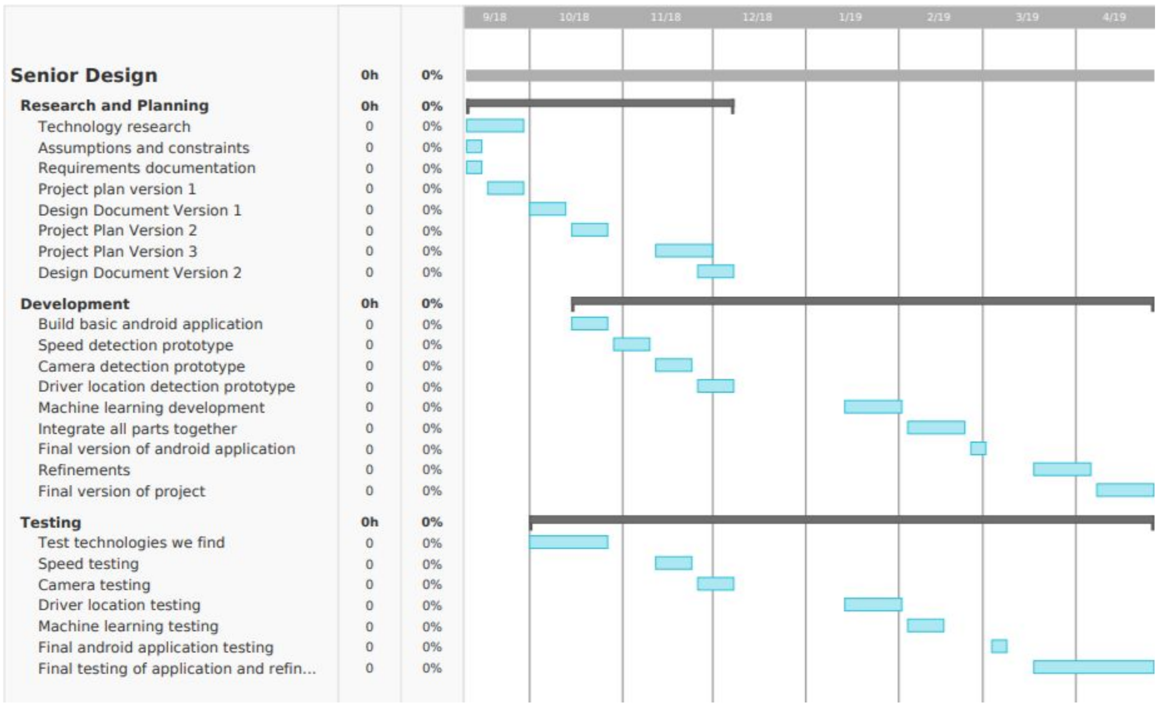

Figure 3: Proposed gantt chart for project

5.3 Risks and Mitigation: Potential vs. Actual

1. When the team was testing, we had to take into consideration the physical health risk of testing an application to detect texting while driving. Our application is able to distinguish between the user sitting on the driver's side or the passenger's side of the car. This allowed one team member to focus on driving while the other members in the car could collect data and test the application.
2. Throughout both semester, the was a scheduling risk. Having six team members meant that finding time to meet for the project was limited since all of us had busy schedules. The project schedule was also very vigorous which did not leave much time for team members to get sick, deal with emergencies, or get injured. We were able to mitigate this risk by making sure multiple team members knew what was going on with different parts of the project so we could pick up work that needed to be completed.
3. Testing required the team to be able to use a car. We were able to use a team members car for testing modules, integration, and the entire application so we avoided the risk of not having a car to be able to test with.

5.4 Lessons learned

Phone Sensor Variability: We learned that different mobile phone devices can produce very different test results. The output readings from the same sensor were consistently different from one mobile device to the next.

Time Constraints: We learned that the breadth of this project would require more time to completely develop, because of the need to learn new technologies such as machine learning.

Hardware Constraints: In the case of inconsistent sensor readings, we learned that the mobile phone is not a viable hardware solution, assuming that we are not using machine learning. An external hardware device that could produce consistent results for all users is required.

Image Processing: We learned that accurate image processing requires a large database of sample photos, upwards of 10,000 to create an accurate classifier.

6. Conclusions

6.1 Closing remarks for the project

Overall this was successful. We determined that it was possible to verify if a user is texting while driving exclusively using available Android sensors and APIs. As this project was being done for research purposes to determine if it possible, we met our main goal. Overall we could have tried to make our application better and have a lower false positive rate, which would have set us up to meet milestone 5, but due to time constraints we were only able to meet milestone 4, of correctly identifying 50% of the time. All source code for the project is available at https://git.ece.iastate.edu/sd/sdmay19-16.

6.2 Future work (potential directions)

The potential for our projects is great as we have laid a solid foundation for project enhancement. The core modules we have implemented successfully show that the values we monitor are capable of determining the position of a phone in vehicle, allowing the application to determine if the user is texting while driving. Our testing revealed that while the application achieves the accuracy we aimed for, long term accuracy can be greatly enhanced by employing the use of machine learning. Machine learning could be used to monitor user texting behaviours on a long term basis, increasing the applications accuracy over time. Furthermore, every phone comes with a different set of sensors with their varying levels of sensitivity which requires different thresholds for modules. Machine learning could customize these thresholds for individual phones by monitoring these sensor values and discrepancies overtime. The future work for this project would benefit greatly with the implementation of machine learning.

List of References

[1] “100 Distracted Driving Facts & Statistics for 2018.” *TeenSafe*, 5 July 2018, www.teensafe.com/distracted-driving/100-distracted-driving-facts-and-statistics-2018/.

[2] “Android Developers.” *Android Developers*, developer.android.com/.

[3] Cheng, Bo, Jian Xuesi, Li Xiang-Yang, and et al. “You’re Driving and Texting: Detecting Drivers using Personal Smart Phones by Leveraging Inertial Sensors.” *Mobicom*. ACM, 2013.

[4] “Drivewise® From Allstate GET REWARDED FOR SAFE DRIVING.” *Allstate*, https://www.allstate.com/drive-wise.aspx

[5] “Safe Drive Zone.” *Safe Drive Zone,* www.safedrivezone.com/.

[6] Wang, Yan, Jie Yang, Hongbo Liu, and et al. “Sensing Vehicle Dynamics for Determining Driver Phone Use.” *MobiSys.* ACM, 2013.

[7] WRAL, “'Oh my gosh!' Raleigh woman's snow photo goes viral,” *WRAL.com*, 13-Feb-2014. [Online]. Available: https://www.wral.com/-oh-my-gosh-raleigh-woman-s-snow-photo-goes-viral/13390109/. [Accessed: 24-Oct-2018].

[8] Bedogni, Luca, Octavian Bujor, Marco Levorato. “Texting and Driving Recognition exploiting subsequent turns leveraging Smartphone sensors.”

Team Information

Kristina Robinson - *Project Lead*
Andrew Knaack - *Lead Designer*
Sara Mace - *Meeting Scribe*
Lucas Golinghorst - *Test Engineer*
Ryan Baker - *Lead Architect*
Derek Clayton - *Report Manager*