Recreating Accelerometer Data In a Simulator While Evaluating Quality Of Driving

Trond Stokkeland

Master's Thesis
Master of Science in Media Technology
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College,

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Recreating Accelerometer Data In a Simulator While Evaluating Quality Of Driving

Trond Stokkeland

2011/06/30

Abstract

Driving simulators are becoming more and more common. For a simulator to be useful it has to be able to recreate events from the real world. This thesis looks at how well we can recreate accelerometer data collected from a simulated vehicle. First we measured accelerometer data from a vehicle driving a set route. Then we processed the data to generate input for our simulator. By using the same sensors we measured how close to reality the g-forces effecting the simulator were.

Finally we organised user testing to recieve feedback on how they felt the simulator works. We asked the testers to evaluate the driving simulation. First by showing them a recorded video. Second by making the simulator move based on the accelerometer data, then finally we showed the users the video while the simulator moved based on the accelerometer data.

Acknowledgments

This master thesis ends my Master in Media Technology study in the Faculty of Computer Science and Media Technology at Gjøvik University College.

First I have to thank my supervisor Simon McCallum for his help with everything from forming the thesis idea to making it happen.

Christer Eriksson from Simuleringssenteret at Høgskolen i Gjøvik deserves thanks for giving me access to the ambulance simulator and teaching me how to use it.

Finally thanks to everyone helping with the testing and experimenting in the ambulance simulator.

Trond Stokkeland, 1st july 2011

Contents

Abstract . . . . **iii**
Acknowledgments . . . . **v**
Contents . . . . **vii**
List of Figures . . . . **ix**
List of Tables . . . . **xi**
1 Introduction **1**
1.1 Topic covered by the Thesis 1
1.2 Keywords 1
1.3 Problem Description 1
1.4 Research Questions 1
1.4.1 Research Question One 1
1.4.2 Research Question Two 1
1.4.3 Research Question Three 2
1.5 Justification, Motivation And Benefits 2
1.6 Methodology 3
1.6.1 Research Question One 3
1.6.2 Research Question Two 3
1.6.3 Research Question Three 3
1.7 Thesis Outline 3
2 Related Work **5**
2.1 Related Work 5
2.2 Driving Quality 5
2.3 Driving Simulators 6
2.4 Entertainment Simulators 6
3 Collecting Data **9**
3.1 The data collected 9
3.2 How data was gathered 9
3.2.1 What an accelerometer does 9
3.3 Programs used 10
3.3.1 Video application 10
3.3.2 Accelerometer Collector 10
3.4 Tools used 10
3.4.1 HTC Desire HD 10
3.4.2 HTC Desire 11
3.4.3 HTC Google Nexus One 11
3.5 Routes 11

3.5.1 Route One . . . 12
3.5.2 Route Two . . . 12
4 Processing Data . . . 15
4.1 The Processing Goals . . . 15
4.2 Reaching The Goals . . . 15
4.2.1 Accelerometer Angle Correction . . . 15
4.2.2 Median Filter . . . 16
4.2.3 From A Size Three Vector To A Single Point . . . 17
4.2.4 Correct Time Intervals . . . 17
4.2.5 Generated Points . . . 17
4.3 Simulator Accelerometer Output . . . 18
5 Experiment With Subjects . . . 21
5.1 The Experiment . . . 21
5.1.1 How it was done . . . 21
5.2 The Simulator . . . 21
5.2.1 Problems . . . 22
5.3 The Interviews & Questionnaire . . . 23
5.3.1 The Interviews . . . 23
5.3.2 The Questionnaire . . . 25
6 Conclusion . . . 27
6.1 Research Question One Conclusion . . . 27
6.2 Research Question Two Conclusion . . . 27
6.2.1 What We Learned From The Interviews . . . 28
6.3 Research Question Three Conclusion . . . 29
7 Future Work . . . 31
7.1 Improvement . . . 31
7.2 Similar Work . . . 32
7.2.1 Car Comfort Comparison . . . 32
7.2.2 Vehicle Lisence Test . . . 32
7.2.3 Movement VS Movie . . . 32
7.2.4 Real Vehicle VS Simulator . . . 32
Bibliography . . . 33
A Video Application Source Code . . . 35
B Accelerometer Application Source Code . . . 41
C Data Processing Source Code . . . 45
D Questionnaire . . . 59

List of Figures

1 The Toyota simulator [1]. 6
2 The VRX iMotion [2]. 7
3 HTC Desire HD attached to the car. 9
4 HTC Desire attached to the car. 9
5 Screenshot of the Accelerometer Collector. 10
6 Map of route one. 12
7 Map of route two. 13
8 Matrix rotation formulas.[3] 15
9 The raw accelerometer data. 16
10 The angle corrected accelerometer data. 16
11 Formula for a vector to a single point. 17
12 The simulator from the rear. 22
13 The hydraulics pump engine. 23
14 The simulator control room. 24
15 Vehicle with three sensors. 31

List of Tables

1 HTC Desire HD Specifications 11
2 HTC Desire Specifications 11
3 HTC Google Nexus One Specifications 12
4 Data statistics of route one. 17
5 Route one point statistics. 17
6 Route two point statistics. 18
7 Route one car data compared to simulator data. 18
8 Route two car data compared to simulator data. 19
9 Results from the questionnaires. 25
10 Results from the questionnaires. 26
11 How realistic does it feel(video and hydraulics), snapshot from 9. 28
12 How realistic does it feel(video and hydraulics), snapshot from 10. 28
13 Driving Quality route one video, snapshot from 9. 29
14 Driving Quality route one video and hydraulics, snapshot from 9. 29
15 Driving Quality route two video, snapshot from 10. 29
16 Driving Quality route two video and hydraulics, snapshot from 10. 30

1 Introduction

1.1 Topic covered by the Thesis

This thesis looks into measuring driving quality using a simulator. We will try to recreate data collected in the real world and assess its quality based on the result.

1.2 Keywords

Smart phone, sensor quality measure, driving quality, simulator.

1.3 Problem Description

Assessing driver quality is a task that is costly and relies in many situations on peoples opinion's. For a computer to assess every possible situation it first has to have prior knowledge of every possible situation that can happen when a vehicle is driven. There have been attempts to make systems for assessing driving, but similar to most systems they also have limitations.

Situations leading to accidents, stunts or other situations that can lead to a car crash can be unsafe and costly to experiment with in real life. Other situations, for example taking a driving license test, can be impractical to reproduce.

Using a simulator can help us solve a variety of problems, but simulators are not without problems. The fact that test subjects know they are safe in a simulator can be enough to influence the outcome of a simulation. How much of reality can we recreate in a fake virtual world, and how much does it influence our test subjects?

1.4 Research Questions

This thesis will answer three questions. The first two questions are the primary questions, the third is a subquestion to question two.

1.4.1 Research Question One

First we will investigate how well a vehicle simulator can be used to recreate a route driven in real life.

- How well can a vehicle simulator recreate results from a recorded drive?

1.4.2 Research Question Two

The data generated in research question one will be used to recreate a vehicle driven in the real world. Then users will sitt in the simulator and evaluate the experience. We do this in order to see if the measured data is perceived as realistic.

- How close to reality does the simulator ride seem for users?

1.4.3 Research Question Three

We will also look at how different the users experience driving quality in the video with and without the hydraulics of the simulator for feedback.

- What effect does a hydraulic system have on the ability to measure driving quality in a video?

1.5 Justification, Motivation And Benefits

Estimating driving quality is a task that can be difficult to control. To control evaluation of driving quality the ability to recreate the route driven is necessary. Under normal circumstances making a vehicle drive a route exactly the same way several times is hard to do. With proper recording we can recreate the driving in a simulator. Car simulators are already in use for driver training in a safe environment, CarSim.com [4]

> ”Driving Simulators are being used worldwide to safely train new car and truck drivers. Pre-accident conditions can be experienced in the simulator allowing new drivers to get a feel for what is safe and what will result in an accident.”[4]

When taking a driving license test the only form of evaluation is the sensor. If the test fails there is no option for appeal or to have the test reevaluated instead, you get a four week waiting period until you can try to take the test again. [5] If the test were to be recorded, and put into a simulator, it would be possible to get the same test reevaluated.

In Norway, 9844 people were injured and 212 people were killed in traffic accidents during 2009 [6]. Driving simulators can help us with both our understanding of how we react in emergencies and help us improve people's driving skill. Driving simulators can provide a safe environment to researchdriving under influence of alcohol, drugs or periods with lack of sleep. In 2010, Riener[7] released a paper on an experiment on people's reaction times while driving. His results show that reaction times were similar in the vehicle and the simulator.

Driving simulators vary a lot in size,shape, and cost. Starting with small home simulators as the one from Olav Sandnes [8] where only two or three projectors are needed together with his software, that cost 100 Euro. Toyota's Sedan simulator [9] is one of the biggest simulators available. It is a dome on rails in a 35 meters x 20 meters hall. As the simulators are getting cheaper, and hence more available, the situations for when we can use a simulator are becoming more frequent.

A direct benefit from this project is that Simuleringssenteret at Høgskolen i Gjøvik is thinking about changing how they use the video and hydraulics input during simulations. If this turns out to be viable it gives them an option to consider.

1.6 Methodology

1.6.1 Research Question One

In order to answer research question one we first need to collect data from a vehicle driving. This will be done by attaching three smart phones to a vehicle. We have developed an Android application that two of the smart phones will use to collect accelerometer data. The third smart phone will use it's camera to record what the driver is seeing through the windscreen.

For the next step we create a processing program to convert the collected accelerometer data into data that can be used as input for the simulator. We will then attach the first two phones in the simulator to gather accelerometer movements. The data processing will be adjusted several times, this will go on until we are able to generate a data set that is close to our original data.

1.6.2 Research Question Two

> ”All research ultimately has a qualitative grounding” - Donald Campbell,[10]
>
> ”There's no such thing as qualitative data. Everything is either 1 or 0” -Fred Kerlinger,[10]

Colorado State University [10] gives a short overview of qualitative research vs quantitative research. While both methods have their flaws selecting one method is often about the researchers preferences and resources.

Together with the best resulting data from research question one we will use the recorded video in the simulator. We went for a qualitative interview with a low number of participants. Some of the participants have been helping with the project at earlier stages and therefore have a fair idea what the project is about. This enables the participants a chance to give feedback on both the projects process and progress.

Questionnaires are commonly used to gather quantitative data, but they can be used as a basis for further in-dept interviews. If it had not been for problems with the ambulance simulator, described in Section 5.2.1, we would have done a combination of qualitative and quantitative research.

1.6.3 Research Question Three

Research question three will be answered at the same time as research question two. While in the simulator, the test subjects will try the route with first only the video. Second with only the hydraulics of the simulator in movement. The test subjects will then run the simulation with both the video and hydraulics at work.

1.7 Thesis Outline

In chapter 2 we look at work related to this thesis. Section 2.2 gives a look at related work in the field of measuring driving quality while section 2.3 goes into driving simulators. We then go to chapter 3. There we can read about what data we need, how we gathered the data and where we gathered it. Chapter 4 is going in detail on the processing of the gathered data. Chapter 5 gives information about the experiment we performed and everything involved. Our conclusion is given in chapter 6. And in the end chapter 7 will give suggestions for improvements and future work to be done.

2 Related Work

2.1 Related Work

2.2 Driving Quality

There are a lot of different reasons to measure driving quality. We have the pay how you drive, found at Newsbeat [11], car insurance where the insurance companies places a black box inside their customer's cars. The insurance companies measure how well people drive and award those who drive well. A goal they also hope to achieve is to make people drive safer and hence lower the amounts of accidents happening on the streets.

> " The Technology
>
> - The smart box is the same size as a mobile phone.
> - It uses satellite technology to track how, and when, young drivers are using their car.
> - It looks at acceleration, braking, cornering and speed, as well as times of journeys.
> - The information is then displayed on the policyholder's online dashboard, where they can log on and get their rating; from five (excellent) to one (very poor).
> - The premium is then re-calculated every 90 days.
> - Driving well can get a discount of up to 11%, driving badly can cost an extra 20%.
>
> " -Newsbeat ,[11]

There are also the personal driving measuring applications, like Dynolicious[12] and DriSMo[13]. DriSMo[13] is an Android application created by a group of students as their bachelor project at Høgskolen i Gjøvik. The way DriSMo[13] works is that it gives you feedback based on the accelerometer data given by your Android phone. They have tested and sat threshold levels for when a reading differs too much from the previous ones.

Goto and Able [14] released in 1995 a paper called "Estimation of driving loci and evaluation of driving skill". Here they describe how to use vehicle's speed and gyro measurements to estimate the road curvature and fluctuating curvature.

> "The spatial spectrum of the fluctuating curvature by driving shows the smoothness of driving, the spatial frequency and amplitude of the driving locus provides a means by which the driver's skill can be estimated." -Goto and Able [14].

Hamada and Nakamori released in 2001 a paper called "Development of In-Vehicle System for Evaluating the "Quality of Driving" " [15], where they describe a system for on-the-fly driving quality evaluation. They connected a camera to a laptop and used image processing to determine placement on the street. This paper is now ten years old so the same technique should be possible to use on a modern smart phone.

While global positioning systems and accelerometer data combined with video are able to collect a lot of the information of how a route is driven, data collection does not stop there. Tateyama and Mori [16] performed two experiments. One where they drove around filming what was going on while tracking eye movement of the driver. In the second experiment they tracked eye movement of people driving around in a virtual world. While driving in the virtual world they also kept track of vehicle steering and position.

2.3 Driving Simulators

Figure 1: The Toyota simulator [1].

Toyota [9] are not the only car manufacturer with a car simulator. General Motors [17] and Ford [18] also have their own simulators. Car manufacturers use simultors to find ways to improve their cars. One experiment is to see how people react in a situation that could lead to a traffic accident, then based on that see how their cars can be improved to prevent the accident or protect the people inside the vehicle.

2.4 Entertainment Simulators

Simulators created for entertainment can sacrifice some realism to be more fun, but they are still worth mentioning.

Simulators for entertainment come in different forms, from the steering wheels and pedals that can be connected to a game console to the larger ones you can find in an amusement park. The VRX iMotion[2] as seen in figure 2 is something in between. To quote the article:

> ”The VRX iMotion is one of the most sophisticated driving simulations in the world, allowing users to drive virtually alongside actual live Nascar and Formula One racers.”- Lewinski [2]

For an estimated sum $30,000, it is available for everyone willing to pay. The simulator works by having access to GPS data from the race cars in actual races. They then put those data to use in a virtual reality and input a model of the real racing cars into the simulator. The VRX iMotion will then give feedback to the user by tilting and vibrating. It is able to generate up to 2G.

Figure 2: VRX iMotion [2].

3 Collecting Data

3.1 The data collected

We gather video of what is going on in front of the vehicle with accurate timestamps from when the video starts and stops. We also gather acceleration forces affecting the vehicle. The acceleration forces consist of gravity from the earth and movement in different directions.

The accelerometer data is received as a size three vector, with an X,Y and Z component representing movement in a 3D-coordinate system. In addition to that we collect accurate timestamps from when the data was measured.

3.2 How data was gathered

Figure 3: HTC Desire HD attached to the car.

Figure 4: HTC Desire attached to the car.

As seen, in Figure 3, the smart phone is in the front of the car recording what the driver is able to see. Figure 4 shows one of the two smart phones connected in the back of the car. There is a smart phone placed directly above each of the back tires. This is done in order to try to get a correct result as possible. We used the HTC Desire on the left side and Nexus One on the right side.

3.2.1 What an accelerometer does

> ” There are many different ways to make an accelerometer! Some accelerometers use the piezoelectric effect - they contain microscopic crystal structures that get stressed by accelerative forces, which causes a voltage to be generated. Another way to do it is by sensing changes in capacitance. If you have two microstructures next to each other, they have a certain capacitance between them. If an accelerative force moves one of the structures, then the capacitance will change. Add some circuitry to convert from capacitance to voltage, and you will get an

> accelerometer. There are even more methods, including use of the piezoresistive effect, hot air bubbles, and light. " -Dimension Engineering[19]

In short an accelerometer measures changes in g-forces (gravitational forces). G-forces work in all directions therefore the accelerometers used in smart phones are 3 axis accelerometers that gives us readings in a 3D space.

3.3 Programs used

3.3.1 Video application

The video application(Listing 1) is an Android application we created to record the video. Basically, it is the same as the standard video recorder coming with the phone. We wrote our own application because we were in need of more accurate timestamps in the video. We need the timestamps to be accurate in order to get correct synchronization with the accelerometer data. The video files are stored in the MPEG-4 Part 14 (MP4) multimedia container format. They are stored with a resolution of 720x480px at 29frames/second. The resolution and framerate was set by default. This happened when we selected the high quality camcorder profile from the Android SDK. With an audio track using 12kbps the total bitrate is 3017kbps. This puts the size of our recorded videos on an average of 85MB.

3.3.2 Accelerometer Collector

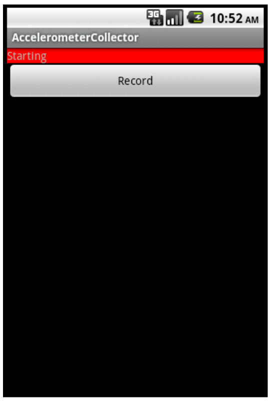

Figure 5: Screenshot of the Accelerometer Collector.

The Accelerometer Collector (Listing 2) is a small application we created that gather all registered accelerometer data. When the application is stopped it writes all collected data to a file. The screenshot in figure 5 shows that we created a very simple interface. The interface tells us the status of the program and enables us to start or stop the program.

3.4 Tools used

3.4.1 HTC Desire HD

From our available smart phones the HTC Desire HD is the one with the best camera. This makes it the best phone to use in front of the car for video recording. The spesifications for the phone

specifications		
Size	Dimensions	123 x 68 x 11.8 mm
	Weight	164 g
Display	Type	LCD capacitive touchscreen, 16M colors
	Size	480 x 800 pixels, 4.3 inches
Memory	Internal	1.5 GB; 768 MB RAM
	Card slot	microSD, up to 32GB
Camera	Video	720p
Features	OS	Android OS, v2.2 (Froyo)
	CPU	1 GHz Scorpion processor, Adreno 205 GPU, Qualcomm MSM8255 Snapdragon
	GPS	Yes, with A-GPS support

Table 1: HTC Desire HD Specifications

can been seen in the table 1.

3.4.2 HTC Desire

specifications		
Size	Dimensions	119 x 60 x 11.9 mm
	Weight	135 g
Display	Type	AMOLED or SLCD capacitive touchscreen, 16M colors
	Size	480 x 800 pixels, 3.7 inches
Memory	Internal	576 MB RAM; 512 MB ROM
	Card slot	microSD, up to 32GB
Camera	Video	WVGA (800x480 pixels) @ 15fps, 720p@30fps via Android 2.2
Features	OS	Android OS, v2.1 (Eclair), upgradeable to v2.2
	CPU	1 GHz Scorpion processor, Adreno 200 GPU, Qualcomm QSD8250 Snapdragon chipset
	GPS	Yes, with A-GPS support

Table 2: HTC Desire Specifications

The HTC Desire is used for collecting accelerometer data together with the HTC Google Nexus One. In terms of hardware they are almost the same. The spesifications for HTC Desire can be seen in the table 2.

3.4.3 HTC Google Nexus One

HTC Google Nexus one is used for collecting accelerometer data together with the HTC Desire. The spesifications for HTC Google Nexus One can be seen in the table 3.

3.5 Routes

When deciding on routes for driving we put down the following criteria:

Duration Should be between three and five minutes.

Variety Contain different traffic situations: round-a-bouts, traffic lights etc.

Local For practical purposes it is convenient to have the routes nearby.

specifications		
Size	Dimensions	119 x 59.8 x 11.5 mm
	Weight	130 g
Display	Type	AMOLED capacitive touchscreen, 16M colors
	Size	480 x 800 pixels, 3.7 inches
Memory	Internal	512MB RAM, 512MB ROM
	Card slot	microSD, up to 32GB
Camera	Video	Yes, D1 (720x480 pixels)@min. 20fps
Features	OS	Android OS, v2.1 (Eclair)
	CPU	1 GHz Scorpion processor, Adreno 200 GPU, Qualcomm QSD8250 Snapdragon chipset
	GPS	Yes, with A-GPS support

Table 3: HTC Google Nexus One Specifications

Reproducable In order to be able to reproduce the results there should not be used any roads with construction work or temporary redirections.

For this thesis we have chosen two short routes that provides a varied in driving experience, and should produce data with diversity.

3.5.1 Route One

Figure 6: Map of the first route driven.

Figure 6 shows a map of route one. Route one is a simple route with four round-a-bouts, speedbumps and speed limits up to 60km/h.

3.5.2 Route Two

The map in figure 7 shows route two. Route two is a route with a steep hill, traffic lights, speed bumps and sharp turns. For most of this route, the speed limit is 30 km/h but increases to 50 km/h for a short amount of time.

Figure 7: Map of the second route driven.

4 Processing Data

4.1 The Processing Goals

What we try to accomplish by processing the data is to make a inout file for the ambulance simulator. That input should make the ambulance simulator generate accelerometer readings close to the data we originally collected from the car.

4.2 Reaching The Goals

In order to achieve the goals for step two we have created a processing program. The processing program (Listing 3) uses the accelerometer data stored by the accelerometer application to output a file that can be used as input for the ambulance simulator. The process involves a set of functions described in detail below.

4.2.1 Accelerometer Angle Correction

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{4.1}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.2}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Figure 8: Matrix rotation formulas.[3]

When collected, the smart phone is not attached in a correct angle. The angle would be correct if the phone stood straight up in a 90 degree angle. In order to correct this a simple matrix rotation is performed. First we rotate the X-axis into its zero position then do the same for the Z-axis. After the rotation the accelerometer data is as if the smart phone were standing with the Y-axis in a 90 degree angle to the earth's gravitational center.

The figures 9and 10 show the results of applying the formulas in figure 8 on the accelerometer data.

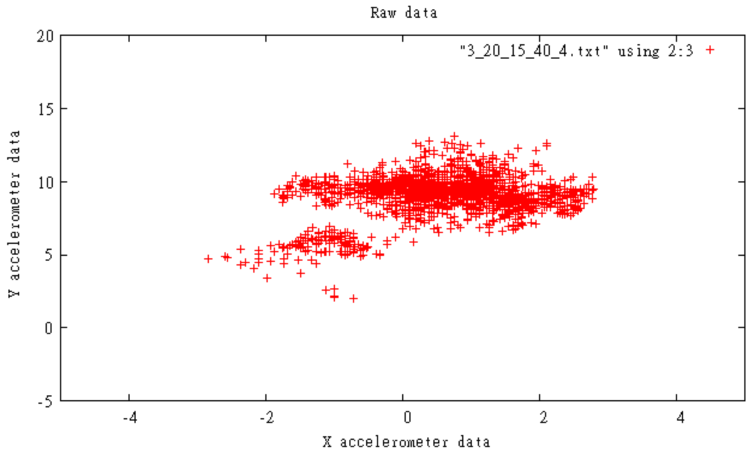

Figure 9: The raw accelerometer data.

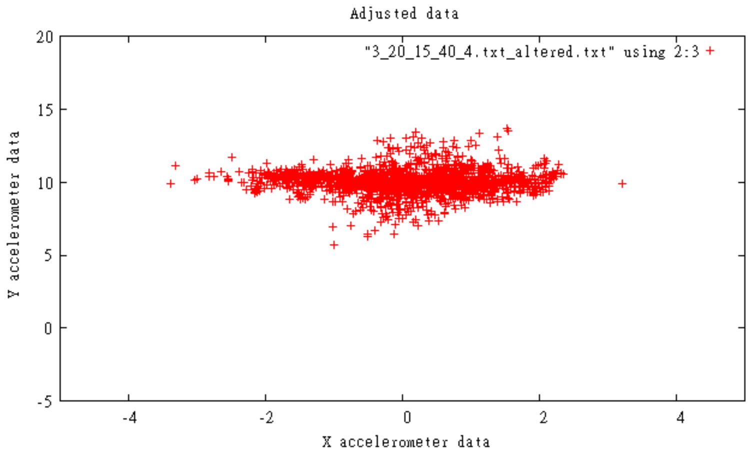

Figure 10: The angle corrected accelerometer data.

4.2.2 Median Filter

> ”For example, suppose that a 3x3 neighborhood has values (10,20,20,20,15,20,20,25,100) . These values are sorted as (10,15,20,20,20,20,20,25,100), witch results in a median of 20. Thus, the principal function of median filters is to force points with distinct intensity levels to be more liker their neighbors.” -Rafael[20]

The book [20] is about image processing, but the theory applies to signal processing. The goal is to get rid of the spikes and give the accelerometer data smoother transitions. The statistics in table 4 shows the effect we get from the median filter. What we see is that it removes potential spikes and the highest and lowest values. The average value remains close to its original value. We use a size five filter because, at times, the spikes appeared in pairs and a size three filter does not remove spikes that are paired. When the spikes appear in pairs, a size three filter will return one of the spiked values.

Data	Left Pre-Filter	Left Post-Filter	Right Pre-Filter	Right Post-Filter
High X	2.7649	2.6014	2.9491	2.9122
Low X	-3.0645	-3.0645	-2.7211	-2.5216
Average X	0.0539	0.0579	-0.0760	-0.0727
High Y	14.5737	13.8355	14.9873	13.4497
Low Y	5.3141	6.8575	5.4704	7.4379
Average Y	10.2635	10.2678	10.4139	10.4124
High Z	4.4780	4.4780	4.5136	4.5136
Low Z	-3.8140	-3.8140	-3.6302	-3.6302
Average Z	-0.0968	-0.0977	0.2884	0.2878

Table 4: Data statistics of route one.

```
double point = accelerometer._x * XMULTIPLYER+accelerometer._y *  YMULTIPLYER + accelerometer._z * ZMULTIPLYER-Original_y + SIMZEROPOINT;
```

Figure 11: Formula for a vector to a single point.

4.2.3 From A Size Three Vector To A Single Point

The simulator uses a simple hydraulics system for moving up and down. The simulator has two hydraulic cylinders, one on each side. The system controlling the pumps takes a single value as input, therefore we need to convert our size three vector of accelerometer data into a single point.

The simulator has a maximum point of 6.0 and a minimum point of -10.0. This gives it about 8 to go on in each direction. If our formula generates a point that is higher, we decrease it down to 6.0 and for points that are lower we increase it up to -10.0. The formula in figure 11 we use is simple. Each axis has a multiplier, and we add inn a SIMZEROPOINT to get the ambulance simulator to be at zero when there is no movement. This is needed because the ambulance simulator stand straight, reciving a signal of -2.2.

4.2.4 Correct Time Intervals

Our accelerometer readings come with a timestamp. Our readings vary frequency, from every 1ms to 170ms. In order to give stable input to the simulator we plot the values and see what the value is every 100ms.

$$Value_n = Value_{n-1} + (\frac{Value_{n+1} - Value_{n-1}}{Time_{n+1} - Time_{n-1}} * Time_n) \tag{4.4}$$

4.2.5 Generated Points

Data	Left sensor	Right sensor
Highest point	6.0	2.8316
Lowest point	-8.0755	-7.64867299404117
Average point	-0.9840	-1.3332

Table 5: Route one point statistics.

Data	Left sensor	Right sensor
Highest point	5.6597	4.1703
Lowest point	-7.8431	-7.7152
Average point	-1.4050	-1.1659

Table 6: Route two point statistics.

The tables 5 and 6 shows statistics of the points generated by the processing program. only the left sensor in route one broke the maximim limit and got pushed down to 6.0. We can also see that we can generate lower points before we hit the minimum imit.

4.3 Simulator Accelerometer Output

We started out with creating different datasets by adjusting the multipliers used in the size three vector to a single point, in Section 4.2.3, method. We then ran the datasets in the simulator while we used the same approach we used for gathering the original accelerometer data. We placed two smart phones in the simulator and let them record the data. After the initial attempt we analyzed the data and adjusted the multipliers based on findings, trying to find values close to the original inout data.

When collecting data from the car we used the HTC Desire on the left side and Nexus One on the right side. In the simulator the Desire was still on the left, but on the right we replaced the Nexus with the Desire HD. The Desire and Desire HD were able to register accelerometer data every twenty miliseconds while the Nexus was only able to register accelerometer data every fourty miliseconds.

Data	Left Car	Left Simulator	Difference	Right Car	Right simulator	Difference
High X	2.6014	1.8474	28.9844%	2.9122	1.2820	55.9783%
Low X	-3.0645	-1.1489	62.5094%	-2.5216	-1.8477	26.7251%
Average X	0.0579	0.2827	388.2556%	-0.0727	-0.3197	339.7524%
High Y	13.8355	13.4925	2.5421%	13.4497	12.7056	5.5325%
Low Y	6.8575	6.8713	0.2012%	7.4379	6.4478	13.3116%
Average Y	10.2678	10.0236	2.3784%	10.4124	9.6352	7.4642%
High Z	4.4780	2.9073	35.0760%	4.5136	2.6423	41.4592%
Low Z	-3.8140	-3.8371	0.6056%	-3.6302	-3.4050	6.2036%
Average Z	-0.0977	-0.6321	546.9805%	0.2878	0.2106	26.8242%
Average sum	10.228	9.6742	5.4146%	10.6275	9.5261	10.3637%

Table 7: Route one car data compared to simulator data.

The data compared in table 7 and table 8 is from after the angle adjustment and median filter was applied.

Data	Left Car	Left Simulator	Difference	Right Car	Right simulator	Difference
High X	2.6518	1.3396	49.4834%	2.6041	1.3491	48.1933%
Low X	-2.9132	-1.3724	52.8903%	-2.9122	-1.6493	43.3659%
Average X	-0.1004	-0.2515	150.4980%	0.0157	0.0367	133.7579%
High Y	15.1454	14.5569	4.0427%	13.9823	12.7244	8.9964%
Low Y	6.9499	7.2519	4.3453%	6.9766	6.8379	1.9881%
Average Y	10.2622	10.0494	2.0737%	10.4195	9.6287	7.5897%
High Z	4.0097	1.9162	52.2109%	4.8009	2.3048	51.9924%
Low Z	-4.3769	-3.1848	27.2362%	-3.2559	-2.8001	13.9993%
Average Z -0.7945		-0.7660	3.5872%	0.6001	-0.0289	104.8158%
Average sum	9.3673	9.0319	3.5806%	11.0353	9.6365	12.6757%

Table 8: Route two car data compared to simulator data.

5 Experiment With Subjects

5.1 The Experiment

5.1.1 How it was done

When the test subjects arrived at the ambulance simulator they got a short briefing about what would happen. They also received the questionnaire so they could start filling out the first page. The first page of the questionnaire was only user information, and it was therefore not important when it got filled out. We then got two and two people to sit together in the simulator. This made the experiment go faster than with only testing one person at a time. It also gave us a chance to see if we got different feedback from people sitting on the left or right side in the simulator. The experiment was split into the six sections :

- Route One Video
- Route One Hydraulics
- Route One Video & Hydraulics
- Route Two Video
- Route Two Hydraulics
- Route Two Video & Hydraulics

After each section the test subjects filled in a form evaluating what they had experienced.

5.2 The Simulator

As can be seen in figure 12, the ambulance simulator is a real ambulance mounted on a platform. The ambulance has been slightly modified with some extra equipment. Sensors have been placed on every door to shut down the ambulance simulator if anyone tries to leave the ambulance simulator during motion. This is a saftey measure done in order to prevent anyone from standing on or near the platform while it is moving. A Logitech steering wheel with pedals and means of communicating with the control room has also been installed in the ambulance simulator.

The ambulance simulator is currently used in training medical personnel. They run a simulation where they drive around, then get dispatched to an accident or similar event. Then they have to drive to the location and perform on site aid, before taking the patient into the ambulance and practice how to treat them in an ambulance while it is in motion.

The Hydraulic System

The ambulance simulator is put in motion by a hydraulic system using two hydraulic cylinders to create movement. The system was built by a bachelor group in 2004 [21].

Figure 12: The simulator from the rear.

Figure 14 show how the control room looks like. From here we use the computer to send data to the hydraulics and display video on the projector in front of the ambulance.

labVIEW

labVIEW [22] is used as the programming interface with the simulator. In this thesis we use an ambulance program previously created for the simulator. One of the program's features is that it can read the data files we generate and send the signals to the ambulance simulator control board.

Responsibility

Responsibility for using the simulator is regulated by "Tivoliloven" (LOV 1991-06-07 nr 24 [23]) (the carnival law).

5.2.1 Problems

The ambulance simulator has for the last six months been having some technical difficulties. In total it has been impossible to use for around three months. This has caused some difficulties with getting all the experiments done, but we have managed to get enough data to analyze.

The engine protection

The engine protection has had some issues with going of and stopping the simulator. It is supposed to protect the engine from overload, but it has been kicking inn at times where everything should be running smooth.

The hydraulic pump

that is in use is not the one the ambulance simulator was built for. The ambulance simulator is supposed to have a pump about three times as powerful as the one currently connected. Because

Figure 13: The hydraulics pump engine.

of this the current pump have problems with maintaining high enough pressure in the hydraulics. Originally the pressure in the hydraulics was 200 bars, now it has been lowered to 120 bars.

5.3 The Interviews & Questionnaire

5.3.1 The Interviews

The interviews were done after the simulator experiment and questionnaire had been filled out. Prior to the interview the questionnaire results were studied and we came up with a list of points we wanted to examine further. The interviews were performed as a conversation about the experiment. This was done in order to give the interview subject a chance to go into topics we had not anticipated. The key points :

- Estimating the driving quality

Figure 14: The simulator control room.

- What felt right or wrong
- Did it feel like sitting in a car
- What could be improved

Sound

One more unexpected comment was that everyone would have liked more feedback in the form of sound. During the experiments the videos had sound, but it was played outside the simulator. Even if it was possible to hear some of it, the simulator has good soundproofing. It is possibly that turning up the volume could help by giving a more realistic experience.

Turns

Another thing everyone agreed on is that the simulator had problems with turning, the sharper the turn was, the less real it felt.

Better before

Three of the test subjects had helped us earlier in the process. In their experience it felt more real before the simulator was adjusted to prevent the engine protection from activating so often.

Noice

At times where the vehicle was supposed to stand still the simulator started to shake a little. One good suggestion we got on this point was to use GPS data to see if the car was in movement or not. Then, if it was standing still simply make the simulator stand still.

Sensitivity

Finding a balance between shaking at the smallest bump and ignoring a gap in the road is hard. Some points gave way too much movements while other places where the subjects expected movement were far too weak.

Video

There was a suggestion that the video could have been processed in a video editor to get rid of the shaking.

The Test Persons

The last two persons who went through the experiment did not have the same experience as the ones before them. The ambulance simulator had been down for a while in between the tests, and now it seem to be in a worse state than before. Both test persons experienced car sickness and felt that the experiment was nowhere near a real driving experience.

5.3.2 The Questionnaire

The questionnaire results have to few entries to do any proper quantitative analysis on them. It is possible to analyse them and speculate on where they might be heading. The test subjects were five male, where all except one currently owns a class B driver license and one female without a class B driver license.

Route One Questionnaire Statistics										
Run One Video										
	1	2	3	4	5	6	7	8	9	10
Driving Quality	0	0	0	0	1	0	2	1	2	0
Video Quality	0	0	1	0	2	0	1	2	0	0
How realistic does it feel	0	2	3	1	0	0	0	0	0	0
Run One Hydraulics										
	1	2	3	4	5	6	7	8	9	10
Driving Quality	0	3	2	1	0	0	0	0	0	0
Does it feel like driving a car	3	1	1	0	1	0	0	0	0	0
Smoothness	1	2	1	2	0	0	0	0	0	0
Run One Video And Hydraulics										
	1	2	3	4	5	6	7	8	9	10
Driving Quality	1	1	1	1	0	0	2	0	0	0
How realistic does it feel	2	0	1	1	1	0	0	1	0	0
Smoothness	2	1	1	1	0	0	1	0	0	0

Table 9: Results from the questionnaires.

Route Two Questionnaire Statistics										
Run Two Video										
	1	2	3	4	5	6	7	8	9	10
Driving Quality	0	0	0	0	1	0	0	2	3	0
Video Quality	0	0	1	1	1	1	1	1	0	0
How realistic does it feel	0	3	2	1	0	0	0	0	0	0
Run Two Hydraulics										
	1	2	3	4	5	6	7	8	9	10
Driving Quality	2	0	1	2	0	1	0	0	0	0
Does it feel like driving a car	2	1	1	0	1	0	0	0	1	0
Smoothness	2	1	1	1	1	0	0	0	0	0
Run Two Video And Hydraulics										
	1	2	3	4	5	6	7	8	9	10
Driving Quality	1	0	1	1	1	1	0	0	1	0
How realistic does it feel	1	1	1	0	2	0	0	0	1	0
Smoothness	2	1	1	1	0	0	1	0	0	0

Table 10: Results from the questionnaires.

6 Conclusion

6.1 Research Question One Conclusion

- How well can a vehicle simulator recreate results from a recorded drive?

In section 4.3 we have a comparison of data collected from the car and the simulator. Both the data from route one in table 7 and the data from route two in table 8 show similar results. The readings on the X and Z axis are very far from each other, with up to 546.9805% difference. The Y axis show much better results. In route one 7 on the left side the highest value has a 2.5421% difference and lowest a 0.2012% difference. On the left side the average is also very good with only a 2.3784% difference. The right side is a bit more of, with a difference of 5.5325% on the high Y and 13.3116% on the low Y, with a average of 7.4642% difference. Since the Y values are so much higher than the X and Z values, the fact that the X and Z values are so far from the original readings does not make that much of an impact. The sum of all average values is only different by 5.4146% on the left side and 10.3637% on the right side.

The numbers for route two 7 are close to those for route one. On the left side, the high Y difference is 4.0427% and the low Y 4.3453% with an average Y difference of 2.0737%. As in route one, the right side of route two also shows a bit worse results. With a high Y difference of 8.9964%, low Y difference of 1.9881% and average Y difference of 7.5897%. The sum of all average values on route two difference is 3.5806% for the left side and 12.6757% for the right side.

The reason for the X and Z axis readings to be so far off the original values can be explained by the simulators limited ability to generate g-forces in those directions. An interesting difference is that the right side shows worse results than the left side. A possible explanation is the different smart phones used, as mentioned in section 4.3. The nexus were only able to get a reading every forty milliseconds while the other smart phones got a reading every twenty millisecond. The problem could be with both the nexus and the Desire HD . It is possible that one of the accelerometer sensors is less accurate or a bit off.

Our conclusion is that the ambulance simulator can match the readings from the car with acceptable difference, but it does have problems with reproducing turns and speed bumps. More research is needed on how to better handle turns, speed bumps and other natural spikes while driving.

6.2 Research Question Two Conclusion

- How close to reality does the simulator ride seem for users?

If we compare results from the question ”How realistic does it feel” for video only and video with hydraulics in table 9, we get that the average answer for video only is 2.83 while for video and hydraulics the average is 3.5. If we remove results from the subjects using the simulator after the bar pressure adjustments our results are 2.75 for the video against 4.0 for video and hydraulics. While our number of test subjects are too low to give any conclusions from a quantitative analysis, we can look at what we have.

The numbers do seem to indicate that the experience in the simulator becomes more realistic with the hydraulics. The numbers from before the latest bar pressure adjustment to the simulator tells us that the realism increased with 45.45% and after with 23.67%.

	1	2	3	4	5	6	7	8	9	10
How realistic does it feel	2	0	1	1	1	0	0	1	0	0

Table 11: How realistic does it feel(video and hydraulics), snapshot from 9.

If we look at the snapshot 11 of table 9 we can see that the results are varied. The average answer was 3.5 if we count all results. But if we only count the people running the experiment while the simulator was more stable we get a average of 4.0.

	1	2	3	4	5	6	7	8	9	10
How realistic does it feel	1	1	1	0	2	0	0	0	1	0

Table 12: How realistic does it feel(video and hydraulics), snapshot from 10.

The snapshot 12 show that the average answer was 4.16 with everyone but 5.5 counting only the first people. Based on only the numbers the simulator is far away from giving its users a realistic feeling of driving. Why this is and how it can be improved we got some answers to during the interviews.

6.2.1 What We Learned From The Interviews

In the interviews ,Section 5.3.1, we noted the key points that could need improvement. Adding more sound might be one of the easiest steps to help it feel more realistic. When people sit in a vehicle they expect to hear the engine, tires against the ground and ambient sounds we normally hear while driving. The ambulance simulator is soundproof and the sounds it makes does not resemble a real vehicle. The room with the simulator has speakers in it, so a possible solution is to turn up the volume and open the simulator windows. The source for the sound would still be misplaced but this is a research area that could be looked into.

Another point that possibly can help with the experience is to edit the video and get rid of the shaking. As a tester said: ”It is normal to shake when sitting in the car, but not that the world around you shakes.”.

Many of the test subjects were concerned that while filling out the questionnaire they did not have a point of reference. In our original planning we did consider to let people drive the selected routes before coming to the simulator. For practical and timing reasons this did not happen.

The people testing the ambulance simulator before the latest adjustments agree that the hydraulics helped making the experience feel more real. They also gave the impression that with some work this could be good.

Our conclusion is that for now it does not feel realistic. Further reasearch on how to improve the experience is needed.

6.3 Research Question Three Conclusion

- What effect does a hydraulic system have on our ability to measure driving quality in a video?

	1	2	3	4	5	6	7	8	9	10
Driving Quality	0	0	0	0	1	0	2	1	2	0

Table 13: Driving Quality route one video, snapshot from 9.

	1	2	3	4	5	6	7	8	9	10
Driving Quality	1	1	1	1	0	0	2	0	0	0

Table 14: Driving Quality route one video and hydraulics, snapshot from 9.

Route one with video only, snapshot 13, give us an average driving quality of 7.5. With the additional feedback of hydraulics, snapshot 14, the average goes down to 4.0.

	1	2	3	4	5	6	7	8	9	10
Driving Quality	0	0	0	0	1	0	0	2	3	0

Table 15: Driving Quality route two video, snapshot from 10.

Video only on route two, snapshot 15, gives an average of 8.0 with hydraulics, snapshot 15, it goes down to 4.66.

The hydraulics alone gives no feeling of driving for the testers. Perceived quality of the driving decreases when the hydraulics are added to the video if we look at the questionnaire results. It can mean that either the driving is worse or the hydraulics makes it a bad experience. Comments from the people who helped us with both making the experiment and the experiment give the impression that when the ambulance simulator is up and running in perfect order, this could

	1	2	3	4	5	6	7	8	9	10
Driving Quality	1	0	1	1	1	1	0	0	1	0

Table 16: Driving Quality route two video and hydraulics, snapshot from 10.

work well. In short, with some more research when the simulator has been repaired this has potential.

7 Future Work

7.1 Improvement

Adding GPS

Using GPS to keep track of the vehicle can provide feedback on current speed.

Synchronice Video With Accelerometer Data

A suggestion about synchronicing the video with the accelerometer data more often. For eksample when you can see a speedbump in the video make sure the acceelrometer data is in the right place. Also when the car is not moving in the video there should not be any movement from the hydrulics either.

A Better Camera

A better camera can help with visual feedback. The current camera on the phone has a limited aperture and therefore provides a narrow image of what is going on.

Three sensors

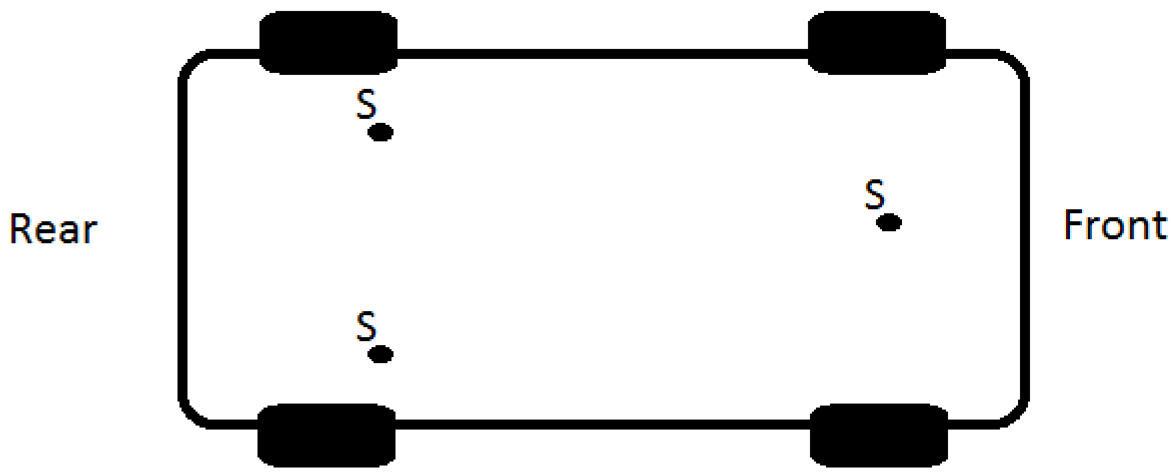

Figure 15: Vehicle with three sensors.

In this thesis we use two smart phones to gather accelerometer data. It is possible to use a third smart phone in the front to get more accelerometer data, and use the data from the third smart phone to adjust the output.

7.2 Similar Work

7.2.1 Car Comfort Comparison

It is possible to collect accelerometer data from different cars driving the same route. Then use the collected data to compare how smooth each of the car drives.

7.2.2 Vehicle Lisence Test

This could be used to record data from driving lisence tests and try to determine out of the data who passed and who failed the test.

7.2.3 Movement VS Movie

Do people trust the movement from the hydraulics or the movie being viewed the most ? This could be testet by driving a route twice. First time with lousy driving, second time with good driving. Afterwards, you take the hydraulics data from the first run and pair it with the video from the second run. It is also possible to take the hydraulics from the second run and video from the first run.

7.2.4 Real Vehicle VS Simulator

Let people drive a given route. While they are driving record the trip. Afterwards, use the collected data in a simulator and let the users compare the experiences.

Bibliography

[1] Roach, A. November 2007. Toyota attempts to reduce accidents, using their new simulator. http://www.automotto.com/entry/toyota-attempts-to-reduce-accidents-using-their-new-simulator/. [Online; accessed june 2011].

[2] Lewinski, J. S. November 2010. Vrx imotion: The first 3d real-time racing simulation. http://www.popularmechanics.com/technology/gadgets/video-games/vrx-imotion-racing-simulation-review?src=rss. [Online; accessed june2011].

[3] Wikipedia. Rotation matrix. http://en.wikipedia.org/wiki/Rotation_matrix. [Online; accessed june 2011].

[4] CarSim. Driving simulators. http://www.carsim.com/products/ds/index.php. [Online; accessed may 2011].

[5] Statens vegvesen. http://www.vegvesen.no/forerkort/foreropplaering/forerproven/generelle+bestemmelser. http://www.vegvesen.no/Forerkort/Foreropplaering/Forerproven/Generelle+bestemmelser. [Online; accessed may 2011].

[6] Statistisk sentralbyr

. 2010. Veitrafikkulykker med personskade. http://www.ssb.no/aarbok/tab/tab-425.html. [Online; accessed may 2011].

[7] Andreas Riener, J. K. 2010. Simulating on-the-road behavior using a driving simulator.

[8] Teknisk Ukeblad. Få din egen simulator hjemme. http://www.tu.no/it/article286077.ece. [Online ; accessed may 2011].

[9] Times, T. J. 2007. Toyota's driving simulator: Not all fun and games. http://search.japantimes.co.jp/cgi-bin/nn20071127a6.html. [Online; accessed may 2011].

[10] Colorado State University. The qualitative versus quantitative debate. http://writing.colostate.edu/guides/research/gentrans/pop2f.cfm. [Online; accessed june 2011].

[11] Newsbeat, B. 5 2011. Pay how you drive. http://www.bbc.co.uk/newsbeat/13549733. [Online; accessed may 2011].

[12] Dynolicious. Dynolicious. http://www.dynolicious.com/. [Online ; accessed june 2011].

[13] Fredrik Hørtvedt, Fredrik Kvitvik, J. A. M. 2011. Drismo-the driving quality application.

[14] Hideyuki Goto, Kyoko Abe, F. M. K. K. K. W. 1995. Estimation of driving loci and evaluation of driving skill. 388.

[15] Yohei Hamada, Takuma Nakamori, N. I. M. N. 2001. Development of in-vehicle system for evaluating the "quality of driving". *Vehicle Electronics Conference, 2001. IVEC 2001.*, 4.

[16] Yoshisuke Tateyama, Yukihiro Mori, K. Y. T. O. H. N. N. K. H. Y. 2010. Car driving behaviour observation using an immersive car driving simulator. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*.

[17] Gary P. Bertollini, Charles M. Johnston, J. W. K. J. C. K. M. A. K. W. E. T. 1994. The general motors driving simulator. In *International Congress & Exposition*.

[18] Jeffry A. Greenberg, T. J. P. 1994. The ford driving simulator. In *International Congress & Exposition*.

[19] Engineering, D. A beginner's guide to accelerometers. `http://www.dimensionengineering.com/accelerometers.htm`. [Online; accessed may 2011].

[20] Rafael C. Gonzales, R. E. W. 2008. *Digital Image Processing*. Prentice-Hall Inc, third edition.

[21] JAN ERIK VOLD, HÅVARD HAARSTAD, T. F. 2004. Styringssystem til simulator.

[22] labVIEW. Product information: What is ni labview? `http://www.ni.com/labview/whatis/`. [Online; accessed may 2011].

[23] Kommunal- og regionaldepartementet. 1991. Tivoliloven. `http://www.lovdata.no/all/hl-19910607-024.html`. [Online; accessed may 2011].

A Video Application Source Code

```
package master.hig.video;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.Calendar;

import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.media.CamcorderProfile;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.os.PowerManager;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;

/*
 * Android application used to capture video and audio, also
 * stores timestamps for when recording starts and stops.
 */
public class VideoRecorder  extends Activity implements OnClickListener, SurfaceHolder.Callback
{

  PowerManager.WakeLock wl;

  MediaRecorder recorder;
    SurfaceHolder holder;
    boolean recording = false;

    //timestamps
    private String _startTime, _stopTime;

    //GPS stuff
    private LocationManager _locationManager;
    private LocationListener _locationListener;
    private String[] _locationData; // TODO : change to arraylist ?
    private int _lastSec = 0;
    private int _counter = 0;
    private String _dataFName;


    private class mylocationlistener implements LocationListener
    {
        @Override
        public void onLocationChanged(Location location)
        {
          if(recording)
          {
            if (location != null)
            {
              Calendar cal = Calendar.getInstance();
              if(_lastSec != cal.get(Calendar.SECOND))
              {
                _lastSec = cal.get(Calendar.SECOND);
                _locationData[_counter] = cal.get(Calendar.HOUR) + ":" +cal.get(Calendar.MINUTE)+
```

```
":"+cal.get(Calendar.SECOND)+":"+cal.get(Calendar.MILLISECOND) +
":" + location.getLatitude() + ":" + location.getLongitude();

                _counter++;
                if(_counter == 48000)
                {
                  recording = false;
                  printGPStoFile();
                }

              }
            }
            }
        }

        @Override
        public void onProviderDisabled(String provider)
        {
        }
        @Override
        public void onProviderEnabled(String provider)
        {
        }
        @Override
        public void onStatusChanged(String provider, int status, Bundle extras)
        {
        }
}



@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags(
WindowManager.LayoutParams.FLAG_FULLSCREEN ,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

    recorder = new MediaRecorder();
    initRecorder();
    setContentView(R.layout.main);

    SurfaceView cameraView = (SurfaceView) findViewById(R.id.CameraView);
    holder = cameraView.getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    cameraView.setClickable(true);
    cameraView.setOnClickListener(this);


    PowerManager pm = (PowerManager)this.getSystemService(this.POWER_SERVICE);
     wl = pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK | PowerManager.ON_AFTER_RELEASE, "VR");
     wl.acquire();

     _locationManager = (LocationManager) getSystemService(this.LOCATION_SERVICE);
     _locationListener = new mylocationlistener();
     _locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,0, _locationListener);
     _locationData = new String[48000];

}

private void printGPStoFile()
{
    File sdCard = Environment.getExternalStorageDirectory();
    File dir = new File (sdCard.getAbsolutePath());
    dir.mkdirs();
    File file = new File(dir, _dataFName);


```

```

    FileOutputStream fOut;
    OutputStreamWriter osw;
  try
  {
    fOut = new FileOutputStream(file);
    osw = new OutputStreamWriter(fOut);

      for(int i=0;i < _counter; i++)
      {
        osw.write(_locationData[i] + "\n");
      }
    osw.flush();
        osw.close();

  }
  catch (Exception e)
  {
    e.printStackTrace();
  }

  _locationData = new String[48000];
}

private void initRecorder()
{
    recorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
    recorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);

    CamcorderProfile cpHigh = CamcorderProfile
            .get(CamcorderProfile.QUALITY_HIGH);
    recorder.setProfile(cpHigh);

    Calendar cal = Calendar.getInstance();

    //setting the name of the recorded file to the current date ,TODO : delete empty unused files..
    String fName = "/sdcard/"+cal.get(Calendar.MONTH)+1+"_"+cal.get(Calendar.DATE)+"_"
+cal.get(Calendar.HOUR_OF_DAY)+"_"+cal.get(Calendar.MINUTE)
+"_"+cal.get(Calendar.SECOND)+".mp4";
    _dataFName = "/sdcard/"+cal.get(Calendar.MONTH)+1+"_"+cal.get(Calendar.DATE)+"_"
+cal.get(Calendar.HOUR_OF_DAY)+"_"+cal.get(Calendar.MINUTE)
+"_"+cal.get(Calendar.SECOND)+".txt";

    recorder.setOutputFile(fName);
    recorder.setMaxDuration(0); //0 = no limit
    recorder.setMaxFileSize(0); // 0 = no limit
}

private void prepareRecorder()
{
    recorder.setPreviewDisplay(holder.getSurface());

    try
    {
        recorder.prepare();
    }
    catch (IllegalStateException e)
    {
        e.printStackTrace();
        finish();
    }
    catch (IOException e)
    {
        e.printStackTrace();
        finish();
    }
}

public void printTimetoFile()
{
  File sdCard = Environment.getExternalStorageDirectory();
    File dir = new File (sdCard.getAbsolutePath() + "/timeData");
    dir.mkdirs();
```

```
    String fname = _startTime + ".txt";
    File file = new File(dir, fname);

    FileOutputStream fOut;
    OutputStreamWriter osw;
  try
  {
    fOut = new FileOutputStream(file);
    osw = new OutputStreamWriter(fOut);

    osw.write(_startTime + "\n");
    osw.write(_stopTime);

    osw.flush();
        osw.close();

  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
}
public void onClick(View v)
{

    if (recording)
    {
        recorder.stop();
        recording = false;


        Calendar cal = Calendar.getInstance();
      _stopTime = cal.get(Calendar.MONTH)+1+"_"+cal.get(Calendar.DATE)+"_"+cal.get(Calendar.HOUR_OF_DAY)
+"_"+cal.get(Calendar.MINUTE)+"_"+cal.get(Calendar.SECOND);

      printTimetoFile();

        printGPStoFile();


        initRecorder();
        prepareRecorder();

    }
    else
    {
        recording = true;
        Calendar cal = Calendar.getInstance();
      _startTime = cal.get(Calendar.MONTH)+1+"_"+cal.get(Calendar.DATE)+"_"+cal.get(Calendar.HOUR_OF_DAY)
+"_"+cal.get(Calendar.MINUTE)+"_"+cal.get(Calendar.SECOND);

        recorder.start();
    }
}

public void surfaceCreated(SurfaceHolder holder)
{
    prepareRecorder();
}

public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
{
}

public void surfaceDestroyed(SurfaceHolder holder)
{
    if (recording)
    {
        recorder.stop();
        recording = false;
    }
    recorder.release();
    wl.release();
```

```
    finish();
}
}
```

Listing 1: Application used to record video and set timestamps.

B Accelerometer Application Source Code

```
package master.hig.accelerometer;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Iterator;

import android.app.Activity;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.os.Environment;
import android.os.PowerManager;
import android.view.SurfaceHolder;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;


/*
 * A android application used to collect accelerometer data during
 *  driving and simulator calibration
 */
public class AccelerometerCollector extends Activity implements SensorEventListener
{

    //accelerometer data
    private SensorManager _sensorManager;
    private Sensor _accelerometer;

    //wake lock
    PowerManager.WakeLock wl;

    private boolean _record = false;
    private ArrayList<String> _accList;
    private Calendar _cal;
    private String _fname;
    private TextView _tv;
    private Button _button;


    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        _tv = (TextView) findViewById(R.id.TextView01);
        _tv.setText("Starting");
        _tv.setBackgroundColor(Color.RED);

        _button = (Button) findViewById(R.id.knapp);
        _button.setText("Record");
        _button.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
              if(_record)
              {
```

```
              stop();
            }
            else
            {
              start();
            }
          }
      });

      PowerManager pm = (PowerManager)this.getSystemService(this.POWER_SERVICE);
       wl = pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK | PowerManager.ON_AFTER_RELEASE , "VR");
       wl.acquire();

      _sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
      _accelerometer = _sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
      _sensorManager.registerListener(this, _accelerometer, SensorManager.SENSOR_DELAY_FASTEST);
       }

  private void start()
  {
    _accList = new ArrayList<String>();
    _record = true;
      _tv.setText("Recording");
    _tv.setBackgroundColor(Color.GREEN );

    _button.setText("Stop");

    _cal = Calendar.getInstance();
    _fname = _cal.get(Calendar.MONTH)+1+"_"+_cal.get(Calendar.DATE)+"_"+_cal.get(Calendar.HOUR_OF_DAY)+"_"
    +_cal.get(Calendar.MINUTE)+"_"+_cal.get(Calendar.SECOND)+".txt";
  }

  private void stop()
  {
    _record = false;
    _tv.setBackgroundColor(Color.RED );
      _tv.setText("Done recording. Saving data.");

      File sdCard = Environment.getExternalStorageDirectory();
      File dir = new File (sdCard.getAbsolutePath() + "/accData");
      dir.mkdirs();
      File file = new File(dir, _fname);

  try {
    FileOutputStream ops = new FileOutputStream(file);

    Iterator<String> itr = _accList.iterator();
      while (itr.hasNext()) {
        String element = itr.next();
        ops.write(element.getBytes());
      }
    ops.flush();
    ops.close();

  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }

  _accList.clear();

      _tv.setText("Data saved. Waiting.");
      _button.setText("Record");



  }

  @Override
public void onAccuracyChanged(Sensor arg0, int arg1)
  {
}
```

```


    public void surfaceDestroyed(SurfaceHolder holder)
    {
        wl.release();
        finish();
    }


  @Override
  public void onSensorChanged(SensorEvent event)
  {

      synchronized (this)
      {

        switch (event.sensor.getType())
        {
          case Sensor.TYPE_ACCELEROMETER:
            if(_record)
            {
                _cal = Calendar.getInstance();
                _accList.add(_cal.get(Calendar.HOUR) + ":" +_cal.get(Calendar.MINUTE)+":"
                    +_cal.get(Calendar.SECOND)+":"+_cal.get(Calendar.MILLISECOND)
                           + "\t" + event.values[0] + "\t" + event.values[1] + "\t"
                           + event.values[2] +"\t");


        }
      }
    }
  }

}
```

Listing 2: Application used to gather accelerometer data.

C Data Processing Source Code

```
/*
 * Small program that handles anything that has to do with reading and processing of data files
 */
public class Datahandler
{
  public static void main(String[] args)
  {
    //Fixer fi = new Fixer("route2_left.txt");
    //Fixer fa = new Fixer("route2_right.txt");
    FileHandler fh = new FileHandler("route2_right.txt",100.0);
    FileHandler fhTwo = new FileHandler("route2_left.txt",100.0);
    Combine com = new Combine("route2_left_altered.txt", "route2_right_altered.txt", 0.1, "route2_3");

  }







}
```

Listing 3: Program used to process data.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Iterator;


/*
 * Class that reads raw accelerometer data and filters and transfer it into
 * input data for the ambulance simulator
 */
public class FileHandler
{
  private final double XMULTIPLYER = 1.0;
  private final double YMULTIPLYER = 1.0;
  private final double ZMULTIPLYER = 1.0;
  private final double POINTMAX = 6.0;
  private final double POINTMIN = -10.0;
  private final double SIMZEROPOINT = -2.2;


  private final double MSPERHOUR = 3600000.0;
  private final double MSPERMIN = 60000.0;
  private final double MSPERSEC = 1000.0;

  private double _timeIntervals; //in MS


  private ArrayList<String> _completeList;
  private ArrayList<Struct> _structList;

```

```
private double _adjust;
private double _adjustTwo;
private boolean _first;
double time = 0.000000;
double _lastTime;


//adjust data
double _y;
boolean _ySet = false;

//time statistics
private double _timeHigh;
private double _timeLow;
private double _timeTotal;
private boolean _firstHandle;

//statistics before median filter
private double _highXValue;
private double _lowXValue;
private double _sumXValue;

private double _highZValue;
private double _lowZValue;
private double _sumZValue;

private double _highYValue;
private double _lowYValue;
private double _sumYValue;

//statistics after median filter
private double _medHighXValue;
private double _medLowXValue;
private double _medSumXValue;

private double _medHighZValue;
private double _medLowZValue;
private double _medSumZValue;

private double _medHighYValue;
private double _medLowYValue;
private double _medSumYValue;

//point statistics
private double _pointHigh;
private double _pointLow;
private double _pointSum;




public FileHandler(String fileName,  double intervals)
{
  _structList = new ArrayList<Struct>();
  _first = true;
  _lastTime = 0.0;
  _firstHandle = true;
  _timeIntervals = intervals;

  //statistics about the time
  _timeHigh = -100.0;
  _timeLow = 100;
  _timeTotal = 0;

  //statistics before median filter
  _highXValue = -100.0;
  _lowXValue = 100.0;
  _sumXValue = 0.0;

  _highZValue = -100.0;
  _lowZValue = 100.0;
  _sumZValue = 0.0;

```

```
    _highYValue = -100.0;
    _lowYValue = 100.0;
    _sumYValue = 0.0;

    //statistics after median filter
    _medHighXValue = -100.0;
    _medLowXValue = 100.0;
    _medSumXValue = 0.0;

    _medHighZValue = -100.0;
    _medLowZValue = 100.0;
    _medSumZValue = 0.0;

    _medHighYValue = -100.0;
    _medLowYValue = 100.0;
    _medSumYValue = 0.0;

    //statistics for the created points
    _pointHigh = -100.0;
    _pointLow = 100.0;
    _pointSum = 0.0;

    _completeList = new ArrayList<String>();



    readFromFile(fileName);
    handleNumbers();
    writeToFile(fileName);

  }


  private void readFromFile(String fileName)
  {
        String tmpStr = "";

        try
        {
          File inFile = new File(fileName);
          BufferedReader br = new BufferedReader(new InputStreamReader(
              new FileInputStream(inFile)));
          while((tmpStr = br.readLine()) != null)
          {
            handleString(tmpStr);
          }


        }
        catch (FileNotFoundException ex)
        {
        }
        catch (IOException ex)
        {
        }
  }

  private void handleNumbers()
  {
    double timepassed = 0.0, p1 = 0.0, p2 = 0.0, value = 0.0, currentTime;
    int valuesAdded = 1, size = _structList.size() ;

    //applying the median filter on all values
    for(int i = 0; i < size; i++ )
    {
      getStats(i, true);
      _structList.get(i).filter(medianFilter(i,5));
      getStats(i, false);
    }

    //start it off at 0.0
    _completeList.add(round(timepassed) + " \t "+ round(structToPoint(_structList.get(0))) + "\n" );

```

```
    for(int i = 1; i < _structList.size()-1; i++)
    {
      timepassed += _structList.get(i)._time;
      currentTime = valuesAdded * _timeIntervals;

      if(timepassed < currentTime )
      {
        if(timepassed + _structList.get(i+1)._time > currentTime)
        {
          p1 = structToPoint(_structList.get(i));
          p2 = structToPoint(_structList.get(i+1));

          value = p1 + ( (p2-p1)/(_structList.get(i+1)._time) * ( currentTime - timepassed ) );
          _completeList.add(round(currentTime ) + "\t"+ round(value) + "\n" );
          valuesAdded++;

        }
      }
      else if(timepassed == currentTime )//it could happen...
      {
        _completeList.add(round(currentTime ) + "\t"+ round(structToPoint(_structList.get(i))) + "\n" );
        valuesAdded++;

      }
      //should only be possible when i = 1;
      else if(timepassed > currentTime )
      {
        if(timepassed - _structList.get(i+1)._time < currentTime)
        {
          p1 = structToPoint(_structList.get(i-1));
          p2 = structToPoint(_structList.get(i));

          value = p1 + ( (p2-p1)/(_structList.get(i)._time-_structList.get(i-1)._time)
              * ( currentTime - timepassed ) );
          _completeList.add(round(_timeIntervals * valuesAdded ) + "\t"+ round(value) + "\n" );
          valuesAdded++;
        }
        else
        {
          //need to add something here to make it move on
          _completeList.add(round(_timeIntervals * valuesAdded ) + "\t"
              + round(structToPoint(_structList.get(i))) + "\n" );
          valuesAdded++;

        }
      }

    }

  }

  private String round(double number)
  {
    String roundedString = "";
    double rounded = 0.0;
    long tmpLong = (int)Math.round(number * 10000);
    rounded = tmpLong / 10000;

    roundedString += rounded;

    return roundedString;
  }


  //metod used to gather statistics about the data before and after applying the median filter
  private void getStats(int pos, boolean beforeMedian)
  {
    Struct s = _structList.get(pos);

    if(beforeMedian)
    {
      // X
      if(s._x > _highXValue)
```

```
    {
      _highXValue = s._x;
    }
    if(s._x < _lowXValue)
    {
      _lowXValue = s._x;
    }

    _sumXValue += s._x;


    //Y

    if(s._y > _highYValue)
    {
      _highYValue = s._y;
    }
    if(s._y < _lowYValue)
    {
      _lowYValue = s._y;
    }

    _sumYValue += s._y;
    //Z

    if(s._z > _highZValue)
    {
      _highZValue = s._z;
    }
    if(s._z < _lowZValue)
    {
      _lowZValue = s._z;
    }

    _sumZValue += s._z;

  }
  else
  {
    // X
    if(s._x > _medHighXValue)
    {
      _medHighXValue = s._x;
    }
    if(s._x < _medLowXValue)
    {
      _medLowXValue = s._x;
    }

    _medSumXValue += s._x;


    //Y

    if(s._y > _medHighYValue)
    {
      _medHighYValue = s._y;
    }
    if(s._y < _medLowYValue)
    {
      _medLowYValue = s._y;
    }

    _medSumYValue += s._y;

    //Z
    if(s._z > _medHighZValue)
    {
      _medHighZValue = s._z;
    }
    if(s._z < _medLowZValue)
    {
      _medLowZValue = s._z;
```

```
      }

      _medSumZValue += s._z;


    }

  }

  //TODO : adjust so this gets right
  private double structToPoint(Struct str)
  {
    double point = (str._x * XMULTIPLYER)+(str._y * YMULTIPLYER)+(str._z * ZMULTIPLYER)-_y+ SIMZEROPOINT;

    if(point > POINTMAX)
    {
      point = POINTMAX;
    }
    if(point < POINTMIN)
    {
      point = POINTMIN;
    }


    if(point > _pointHigh)
    {
      _pointHigh = point;
    }
    if(point < _pointLow)
    {
      _pointLow = point;
    }

    _pointSum += point;

    return point;
  }

/**
 * method used to print out the processed data and the statistics about the data.
 */
  private void writeToFile(String fileName)
  {
    String[] parts = fileName.split("\\.");
    String changedFileName = parts[0] + "_altered.txt";

    File outFile = new File(changedFileName);
    try {
      FileOutputStream ops = new FileOutputStream(outFile);

      Iterator<String> itr = _completeList.iterator();
        while (itr.hasNext()) {
          String element = itr.next();
          ops.write(element.getBytes());
        }
      ops.flush();
      ops.close();

    } catch (FileNotFoundException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }

    String statFileName = parts[0] + "_statistics.txt";
    outFile = new File(statFileName);
    try {
      FileOutputStream ops = new FileOutputStream(outFile);
      String tmp = "Time statistics \n";
      tmp += "Readings: " + _structList.size() + "\n TimeHigh: " + _timeHigh + "\n TimeLow: " + _timeLow
      + "\n TimeTotal: \t" + _timeTotal + "\n Time Average: " + (_timeTotal / _structList.size()) +" \n";
      ops.write(tmp.getBytes());
      tmp = "PreFilter statistics \n";
```

```
      tmp += " Highest X: \t " + _highXValue + " \n Lowest X: \t " + _lowXValue + " \n Sum X: " + _sumXValue
      + " \n Average X: \t" + (_sumXValue / _structList.size()) + "\n";
      tmp += " Highest Y: \t " + _highYValue + " \n Lowest Y: \t " + _lowYValue + " \n Sum Y: " + _sumYValue
      + " \n Average Y: \t" + (_sumYValue / _structList.size()) + "\n";
      tmp += " Highest Z: \t " + _highZValue + " \n Lowest Z: \t " + _lowZValue + " \n Sum Z: " + _sumZValue
      +" \n Average Z: \t" + (_sumZValue / _structList.size()) + "\n";
      ops.write(tmp.getBytes());
      tmp = "PostFilter statistics \n";
      tmp += " Highest X: \t " + _medHighXValue + " \n Lowest X: \t " + _medLowXValue + " \n Sum X: "
      + _medSumXValue + " \n Average X: \t" + (_medSumXValue / _structList.size()) + "\n";
      tmp += " Highest Y: \t " + _medHighYValue + " \n Lowest Y: \t " + _medLowYValue + " \n Sum Y: "
      + _medSumYValue +" \n Average Y: \t" + (_medSumYValue / _structList.size()) + "\n";
      tmp += " Highest Z: \t " + _medHighZValue + " \n Lowest Z: \t " + _medLowZValue + " \n Sum Z: "
      + _medSumZValue +" \n Average Z: \t" + (_medSumZValue / _structList.size()) + "\n";
      ops.write(tmp.getBytes());
      tmp = "Point statistics \n";
      tmp += "Highest point: \t" + _pointHigh + " \n Lowest point: \t" + _pointLow + " \n Average point:\t"
      + (_pointSum / _structList.size()) + "\n";
      ops.write(tmp.getBytes());
      ops.flush();
      ops.close();

    } catch (FileNotFoundException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }



  }



  private Struct medianFilter(int pos, int size)
  {
    int halfSize = size/2;

  Struct[] numbers = new Struct[size];

    if(pos-2 >= 0)
    {
      numbers[0] = _structList.get(pos-2);
    }
    else
    {
      numbers[0] = _structList.get(pos);
    }
    if(pos-1 >= 0)
    {
      numbers[1] = _structList.get(pos-1);
    }
    else
    {
      numbers[1] = _structList.get(pos);
    }
    numbers[2] = _structList.get(pos);
    if(pos+1 >= _structList.size())
    {
      numbers[3] = _structList.get(pos);
    }
    else
    {
      numbers[3] = _structList.get(pos+1);
    }
    if(pos+2 >= _structList.size())
    {
      numbers[4] = _structList.get(pos);
    }
    else
    {
      numbers[4] = _structList.get(pos+2);
    }
```

```

    Arrays.sort(numbers);

    return numbers[halfSize];
  }


  private void handleString(String str)
  {
    String[] parts = str.split("\t"); //0 = time, 1 = X, 2 = Y, 3 = z
    Double[] values = new Double[3];// 0 = x, 1 = y, 2 = z
    String[] divTime = parts[0].split(":");
    Double hour = Double.valueOf(divTime[0]);
    Double min = Double.valueOf(divTime[1]);
    Double sec = Double.valueOf(divTime[2]);
    Double ms = Double.valueOf(divTime[3]);


    //to calculate the time between the readings
    Double thisTime = (hour * MSPERHOUR ) + (min * MSPERMIN) + (sec * MSPERSEC) + ms;
    Double timeDif;
    if(_firstHandle)
    {
      timeDif = 0.0;
      _firstHandle = false;
    }
    else
    {
      timeDif = thisTime - _lastTime;
      _timeTotal += timeDif;

      if(timeDif > _timeHigh)
      {
        _timeHigh = timeDif;
      }
      if(timeDif < _timeLow && timeDif != 0.0)
      {
        _timeLow = timeDif;
      }

    }
    _lastTime = thisTime;




    for(int i =0; i <3; i++)
    {
      values[i]= Double.valueOf(parts[i+1].trim()).doubleValue();
    }

      double _sum = (values[0]*values[0]) + (values[1]*values[1]) + (values[2]*values[2]);
      double _sqrt = Math.sqrt(_sum);
      double _iSinZ = values[2]/_sqrt;
      double _aSinZ = Math.asin(_iSinZ);
      if(_first)
      {
      _adjust = _aSinZ * (180/Math.PI);

      }
      double _angle = (_aSinZ * (180/Math.PI) ) - _adjust;

      values[1] = _sqrt * (Math.cos(Math.toRadians(_angle)));
      values[2] = _sqrt * (Math.sin(Math.toRadians(_angle)));


      double _iSinX = values[0]/_sqrt;
      double _aSinX = Math.asin(_iSinX);
      if(_first)
      {
      _adjustTwo = _aSinX * (180/Math.PI);
      _first = false;
      }
```

```
      double _angleTwo = (_aSinX * (180/Math.PI) ) - _adjustTwo;

      values[1] = _sqrt * (Math.cos(Math.toRadians(_angleTwo)));
      values[0] = _sqrt * (Math.sin(Math.toRadians(_angleTwo)));

      if(!_ySet)
      {
        _ySet = true;
        _y = values[1];
      }

    _structList.add(new Struct(values[0],values[1], values[2], timeDif));
  }
}
```

Listing 4: Class that handles reading and main processing of data.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;


/*
 * Class used to combine two sets of simulator input data into a single file
 * if for some reason one file is longer than the other the shortest file will decide the length
 */
public class Combine
{
  private String[] _dataOne;
  private String[] _dataTwo;
  private int _counterOne;
  private int _counterTwo;
  private double _interval;

  public Combine(String file1, String file2, double interval, String outputFileName)
  {
    _dataOne = new String[48000]; //TODO : change to arraylist ?
    _dataTwo = new String[48000]; // TODO : change to arraylist ?
    _counterOne = 0;
    _counterTwo = 0;
    _interval = interval;

    readFromFile(file1,1);
    readFromFile(file2,2);
    combineFiles(outputFileName);

  }

  private String numbers(Double number)
  {
    String test = number.toString();
    String[] parts = test.split("\\.");
    if(parts[1].length() > 6)
    {
      return parts[0] + "." + parts[1].substring(0,5);
    }
    else
    {
      return test;
    }
  }

  private String numbers(String number)
  {
    String[] parts = number.split("\\.");
    if(parts[1].length() > 6)
    {
```

```
      return parts[0] + "." + parts[1].substring(0,5);
    }
    else
    {
      return number;
    }
  }




  private void readFromFile(String fileName, int nr)
  {
        String tmpStr = "";

        try
        {
          File inFile = new File(fileName);
          BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(inFile)));
          while((tmpStr = br.readLine()) != null)
          {
            if(nr == 1)
            {
              _dataOne[_counterOne] = tmpStr;
              _counterOne++;

            }
            else
            {
              _dataTwo[_counterTwo] = tmpStr;
              _counterTwo++;
            }
          }


        }
        catch (FileNotFoundException ex)
        {
        }
        catch (IOException ex)
        {
        }
  }

  private void combineFiles(String fileName)
  {
  //  String[] parts = fileName.split("_|\."); //TODO: Fix the split
  //  String newName = parts[0] + "_combined.txt";
  //  newName = "testCombined";
    int counter = 0;

    if(_counterOne < _counterTwo)
    {
      counter = _counterOne;
    }
    else
    {
      counter = _counterTwo;
    }

    String header = "LabVIEW Measurement\n" +
            "Writer_Version\t2\n" +
            "Reader_Version\t2\n" +
            "Separator\tTab\n" +
            "Decimal_Separator\t.\n" +
            "Multi_Headings\tNo\n" +
            "X_Columns\tMulti\n" +
            "Time_Pref\tAbsolute\n" +
            "Operator\tambulanse\n" +
            "Date\t2011/04/11\n" +
            "Time\t19:32:55,234375\n" +
            "***End_of_Header***\n\n"  +
            "Channels\t2\n" +
```

```
            "Samples\t1\t1\n"+
            "Date\t2011/04/11+\t2011/04/11\n" +
            "Time\t19:32:55,234375\t19:32:55,234375\n" +
            "X_Dimension\tTime\tTime\n" +
            "X0\t0,0000000000000000E+0\t0,0000000000000000E+0\n" +
            "Delta_X\t1,000000\t1,000000\n" +
            "***End_of_Header***\n" +
            "X_Value\tUntitled\tX_Value\tUntitled 1\tComment\n";

    File outFile = new File(fileName);

    try
    {
      FileOutputStream ops = new FileOutputStream(outFile);
      ops.write(header.getBytes());
      String tmp = "";
      String[] dataOneParts;
      String[] dataTwoParts;
      for(int i = 0; i< counter; i++)
      {
        dataOneParts = _dataOne[i].split("\t");
        dataTwoParts = _dataTwo[i].split("\t");

        tmp = numbers((double)i * _interval ) + "\t" + numbers(dataOneParts[1])
 + "\t" + numbers((double)i * _interval ) + "\t" + numbers(dataTwoParts[1]) + "\n";
        ops.write(tmp.getBytes());
      //  ops.write(_fixedAgain[i].getBytes());
      }
      ops.flush();
      ops.close();

    }
    catch (FileNotFoundException e)
    {
      e.printStackTrace();
    }
    catch (IOException e)
    {
      e.printStackTrace();
    }
  }
}
```

Listing 5: Class that takes two input files and converts them to a single file witch the simulator can use.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Iterator;


/*
 * small class made to fix files from the accelerometerApp that were missing a \n per reading and made the
 * files a single long line
 */
public class Fixer {
  private ArrayList<String> _fixedList;


  public Fixer(String fileName)
  {
    _fixedList = new ArrayList<String>();
    readFromFile(fileName);
    writeToFile(fileName);
```

```
}


private void readFromFile(String fileName)
{
      String tmpStri = "", tmpStr;
      int tabCount = 0;

      try
      {
        File inFile = new File(fileName);
        BufferedReader br = new BufferedReader(new InputStreamReader(
            new FileInputStream(inFile)));
        while((tmpStr = br.readLine()) != null)
        {
            for(String s : tmpStr.split("\\t"))
            {

              if(tabCount < 3)
              {
              tmpStri += s +"\t";
              tabCount++;
              }
              else if(tabCount == 3)
              {
                tmpStri += s + "\n";
                tabCount = 0;
                _fixedList.add(tmpStri);
                tmpStri = "";
              }


            }


        }


      }
      catch (FileNotFoundException ex)
      {
      }
      catch (IOException ex)
      {
      }
}

private void writeToFile(String fileName)
{
  String[] parts = fileName.split("\\.");
  String changedFileName = parts[0] + "_altered.txt";


  try {
    FileOutputStream ops = new FileOutputStream(fileName);

    Iterator<String> itr = _fixedList.iterator();
      while (itr.hasNext()) {
        String element = itr.next();
        ops.write(element.getBytes());
      }
    ops.flush();
    ops.close();

  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
```

```
}
}
```

Listing 6: Class made to fix a bug in the latest accelerometer application app.

```
/*
 * Class to store accelerometer data
 */
public class Struct implements Comparable<Struct>
{
  double _x,_y,_z,_time;


  public Struct()
  {

  }

  public Struct(double x, double y, double z, double time)
  {
    _x = x;
    _y = y;
    _z = z;
    _time = time;
  }

  //adding the values while keeping the timestamp
  public void filter(Struct struct)
  {
    this._x = struct._x;
    this._y = struct._y;
    this._z = struct._z;
  }

  public void setX(double x)
  {
    _x = x;
  }
  public void setY(double y)
  {
    _y = y;
  }
  public void setZ(double z)
  {
    _z = z;
  }


  //Comparing by Y because it is the value that varies the most.
  @Override
  public int compareTo(Struct str) {
    if(this._y < str._y )
    {
      return -1;
    }
    if(this._y > str._y)
    {
      return 1;
    }

    return 0;
  }


}
```

Listing 7: Class used to store accelerometer data.

D Questionnaire

Driving Quality In Simulator

There are 16 questions in this survey

User Information

1 [1.1]Gender : *

Please choose **only one** of the following:

- Female
- Male

2 [1.2]Age : *

Please write your answer here:

3 [1.3]Do you own a driver licence ? *

Please choose **only one** of the following:

- Yes
- No

4 [1.4]Comments :

Please write your answer here:

Run One Video

5 [2.1]On a Scale 1-10 (where 10 is the best) rate the following qualities

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Driving Quality	○	○	○	○	○	○	○	○	○	○
Video Quality	○	○	○	○	○	○	○	○	○	○
How realistic does it feel	○	○	○	○	○	○	○	○	○	○

6 [2.2]Comments :

Please write your answer here:

Run One Hydraulics

7 [3.1]On a Scale 1-10 (where 10 is the best) rate the following qualities

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Does it feel like driving	○	○	○	○	○	○	○	○	○	○
Driving Quality	○	○	○	○	○	○	○	○	○	○
Smoothness	○	○	○	○	○	○	○	○	○	○

8 [3.2]Comments :

Please write your answer here:

Run One Video And Hydraulics

9 [4.1]On a Scale 1-10 (where 10 is the best) rate the following qualities

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Driving Quality	○	○	○	○	○	○	○	○	○	○
How realistic does it feel	○	○	○	○	○	○	○	○	○	○
Smoothness	○	○	○	○	○	○	○	○	○	○

10 [4.2]Comments :

Please write your answer here:

Run Two Video

11 [5.1]On a Scale 1-10 (where 10 is the best) rate the following qualities

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Driving Quality	○	○	○	○	○	○	○	○	○	○
Video Quality	○	○	○	○	○	○	○	○	○	○
How realistic does it feel	○	○	○	○	○	○	○	○	○	○

12 [5.1]Comments :

Please write your answer here:

Run Two Hydraulics

13 [6.1]On a Scale 1-10 (where 10 is the best) rate the following qualities

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Does it feel like driving	○	○	○	○	○	○	○	○	○	○
Driving Quality	○	○	○	○	○	○	○	○	○	○
Smoothness	○	○	○	○	○	○	○	○	○	○

14 [6.2]Comments :

Please write your answer here:

Run Two Video And Hydraulics

15 [7.1]On a Scale 1-10 (where 10 is the best) rate the following qualities

Please choose the appropriate response for each item:

	1	2	3	4	5	6	7	8	9	10
Driving Quality	○	○	○	○	○	○	○	○	○	○
How realistic does it feel	○	○	○	○	○	○	○	○	○	○
Smoothness	○	○	○	○	○	○	○	○	○	○

16 [7.2]Comments :

Please write your answer here: