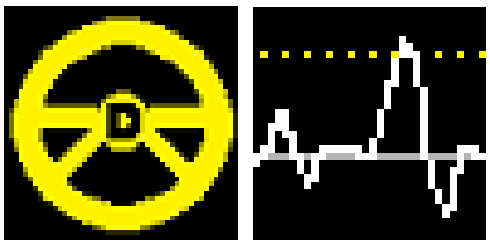
ECE1778 – Creative Applications for Mobile Devices

Final Report – DriveMod



Frances Awachie, Student 992915976

Adrian Matheson, Student 999213239

Matthew Thorpe, Student 995395889

Submitted 2012-04-12

Total word count: 2498

Apper context word count: 500

Table of Contents

1. Introduction

The DriveMod mobile application was created to help individuals modify their driving performance and to enable research into modifying the task as a whole for all drivers, in both cases working toward better safety and efficiency.

The app was developed by two graduate students in Electrical and Computer Engineering and one in Industrial Engineering. The more specific field of the third team member, or the 'apper' of the group, is Cognitive Human Factors Engineering, the focus of which is ensuring that any human-machine system has the specific mental strengths and weaknesses of humans well accounted for in the design of the rest of the system.

The task of operating a motor vehicle was seen as ideal for adding a personal mobile device to the 'machine'. It was specifically chosen for intervention in this way because:

a) this task is carried out by many people, very often;
b) it can have grave consequences for human health—In 2009, Canadians suffered 2209 fatalities and 172 883 injuries due to motor vehicle collisions[1];
c) given a) and b), in this area even small performance improvements are valuable;
d) performance can be improved with instant and delayed feedback; and
e) some of the related technology and other 'machine' elements in the complete driving system evolve slowly (basic vehicle user interface, roadway construction), so an app represents a vector for uncharacteristically rapid advancement.

Perhaps even more important than being able to offer drivers better detection of instances of poor driving than they could manage on their own, a phone, being compact and rich in sensors and data recording capability, was seen as having excellent potential as a data collection tool for research on this inherently mobile task.

2. Overall Design

Safety is the predominant concern for this app, but the definition of 'poor' driving for its purposes includes fuel inefficiency. This is primarily because it was trivial to add the capability to detect it, but also because inefficient driving is linked with speed, which is in turn implicated in many vehicle accidents. The core function of the app is to identify the following accelerometric 'events' for the following reasons:

a) Abrupt Turning – This is strong lateral acceleration for a relatively short time and is indicative of momentary inattention.
b) Hard Turning – This is medium-strong lateral acceleration for a relatively long time and is indicative of a driver carrying too much speed through a curve or while turning at an intersection.
c) Abrupt Braking – This is strong rearward acceleration for a relatively short time and is indicative of momentary inattention.
d) Hard Braking – This is a medium-strong rearward acceleration for a relatively long time and is indicative of the driver having too short of a following distance given the speed of travel
e) Hard Throttle – This is a medium-strong forward acceleration for a relatively long time and is indicative of being hard on the throttle and thus fuel-inefficient.

The app also collects many types of data that could be the subject of research into the causes of poor driving including time, location, and ambient audio.

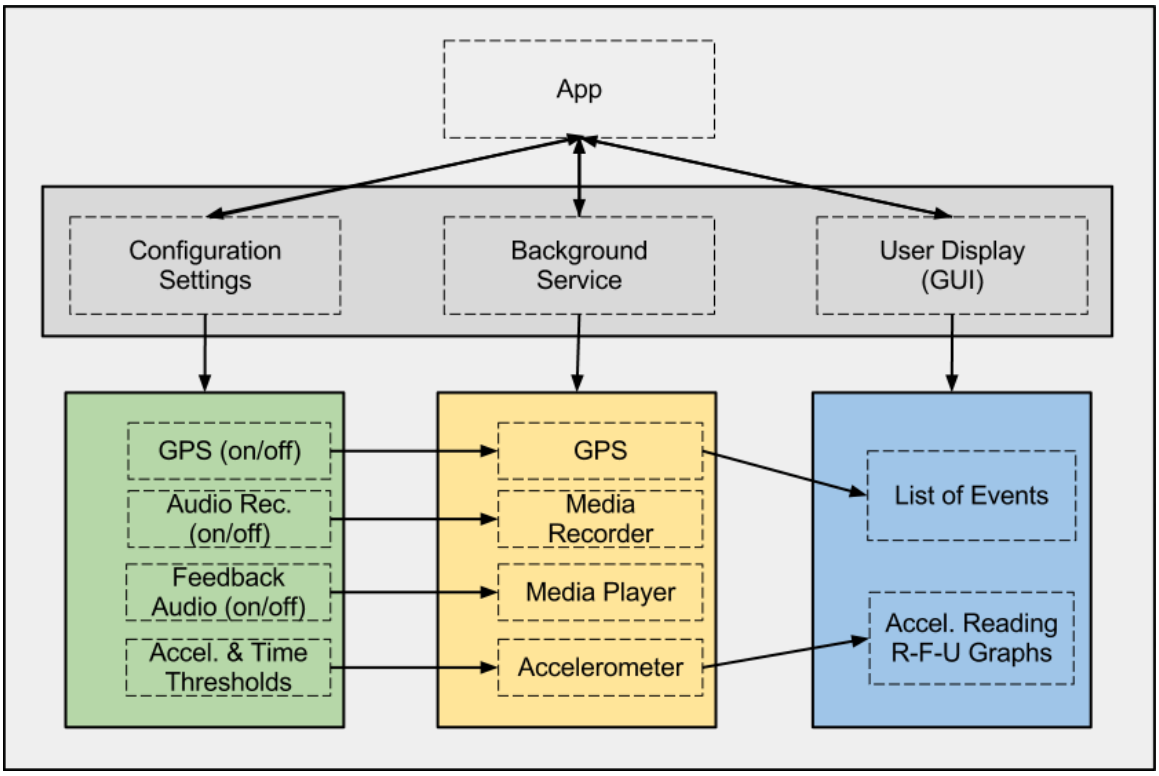


Figure 1: Block diagram of overall DriveMod application

Figure 1 shows the block diagram of the overall application. The Configuration settings deal with location determination (GPS), audio recording, audio feedback, and event settings. Aside from the acceleration and time span thresholds for the different events, most of these are on-off controls. They allow the user to determine what features they want to be tracked, saved, or played during the next trip. With GPS 'on' the app saves frequent GPS (or Wi-Fi) location data and flags those related to events. With audio recording turned on, it records a continuous series of clips and flags those at most 20 seconds before to 10 seconds after detected events (similar to an aviation black box). With audio feedback turned on, the app gives the user audible messages announcing (after a short delay, so as not to add distraction at a crucial moment) when an event has been detected.

The background service houses all data recording (input of location, audio, and accelerations), event detection and related calculations, and feedback (audio output).

The User Display includes all user interface elements required to set up, launch, and stop trip data collection, shows information from any previous trip, and allows the user to back up all app trip data to a file.

Figure 2 presents details of the DriveMod elements specifically dealing with accelerations. At the beginning of every trip, the app must determine the orientation of the vehicle with respect to the phone since all 'events' are defined by accelerations along the vehicle's axes rather than the phone's. All incoming accelerometer values must be converted from one co-ordinate system to the other before event-detection logic is applied. This involves learning the direction of the 'down' direction through the constant force of gravity, the learning of 'forward' direction through the initial movement of the vehicle, and vector algebra involving cross-products to generate an alternate axis system.

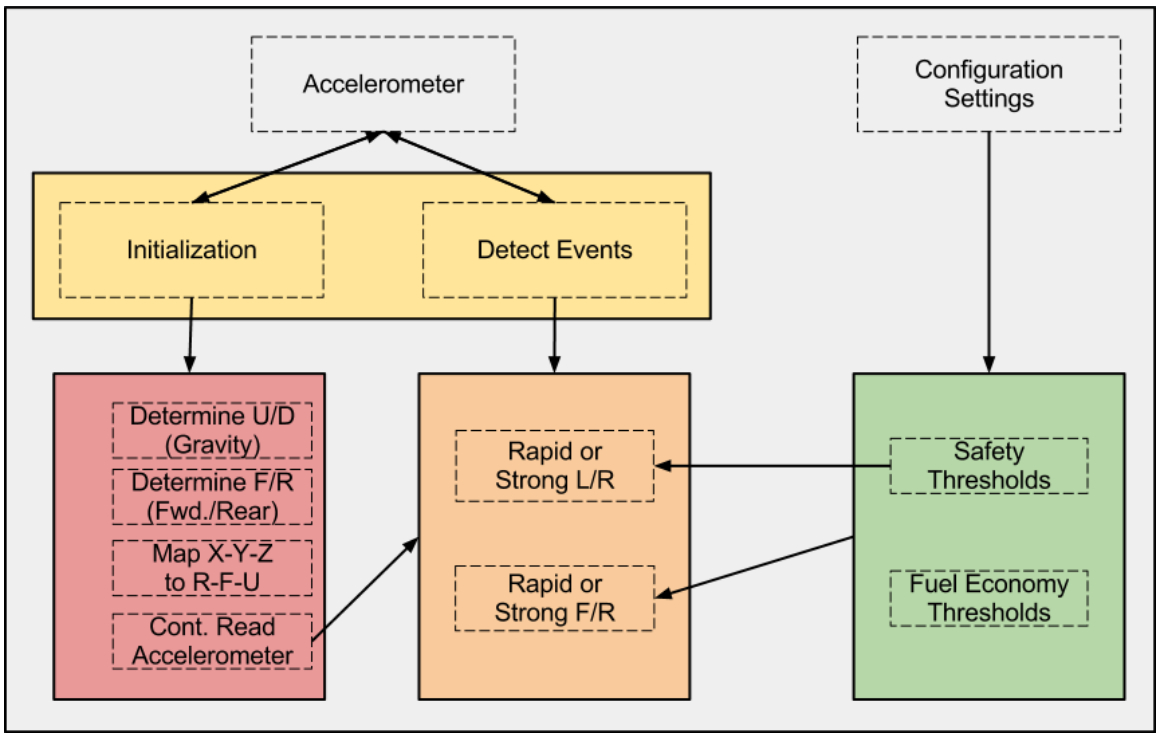


Figure 2: Block diagram of accelerometer-related details of DriveMod application

In addition to the poor driving events, the app detects rough roads via detecting major vertical accelerations. At these times flagging of all events is temporarily suspended as they would likely be false positives caused by the roughness of the road. When an event is flagged, a database entity including an event Id number, the instantaneous R-F-U accelerometer readings, the event time, and the location (settings-dependent) is created. Audio feedback (settings-dependent) and toast, both describing the exact event type, are also presented, though the latter is solely for program testing.

3. Functionality

The core of the application is its ability to generate a vehicle-relative (R,F,U) co-ordinate system, to convert the conventional (X,Y,Z) system accelerometer values into our (R,F,U) system, and to detect events. The app carries out orientation detection accurately, it detects all five of our events under the proper conditions, and it provides the proper audio feedback cues following events. Detection is also correctly suppressed under the conditions specified in our rough road rejection function, and the app adheres the user settings mandating minimum accelerometer sample spacing and minimum time between the flagging of events of the same type. The screenshots in Figures 3 and 4 relate to these functions.

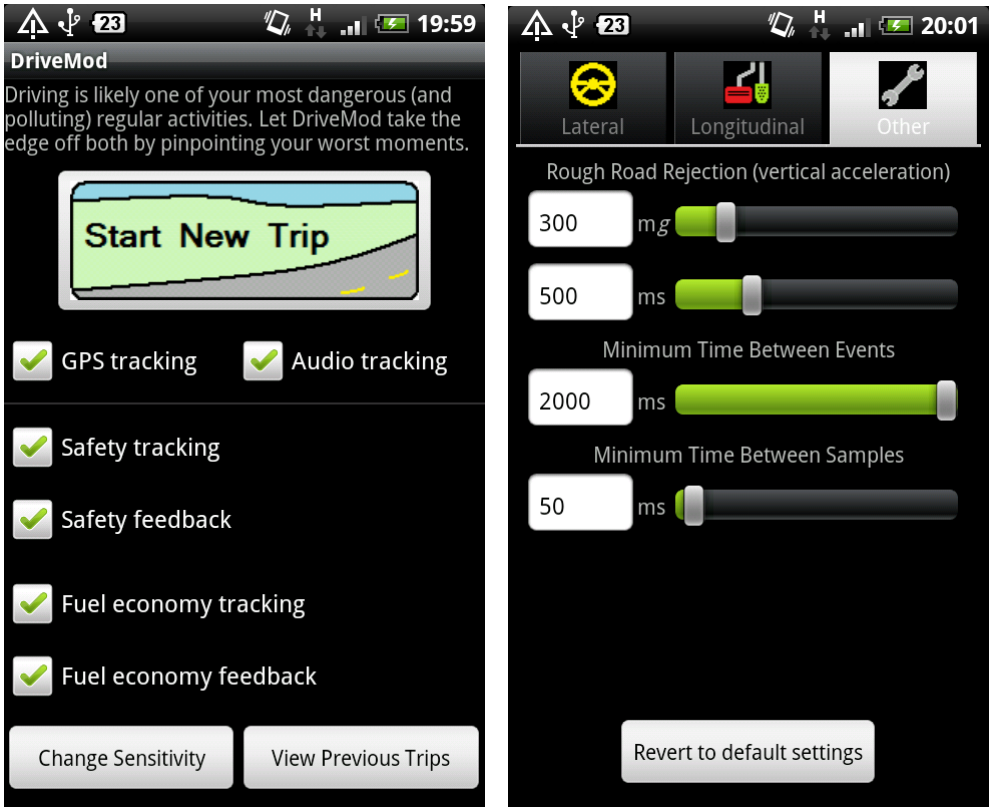


Figure 3: DriveMod main screen; DriveMod Settings activity showing 'Other' tab

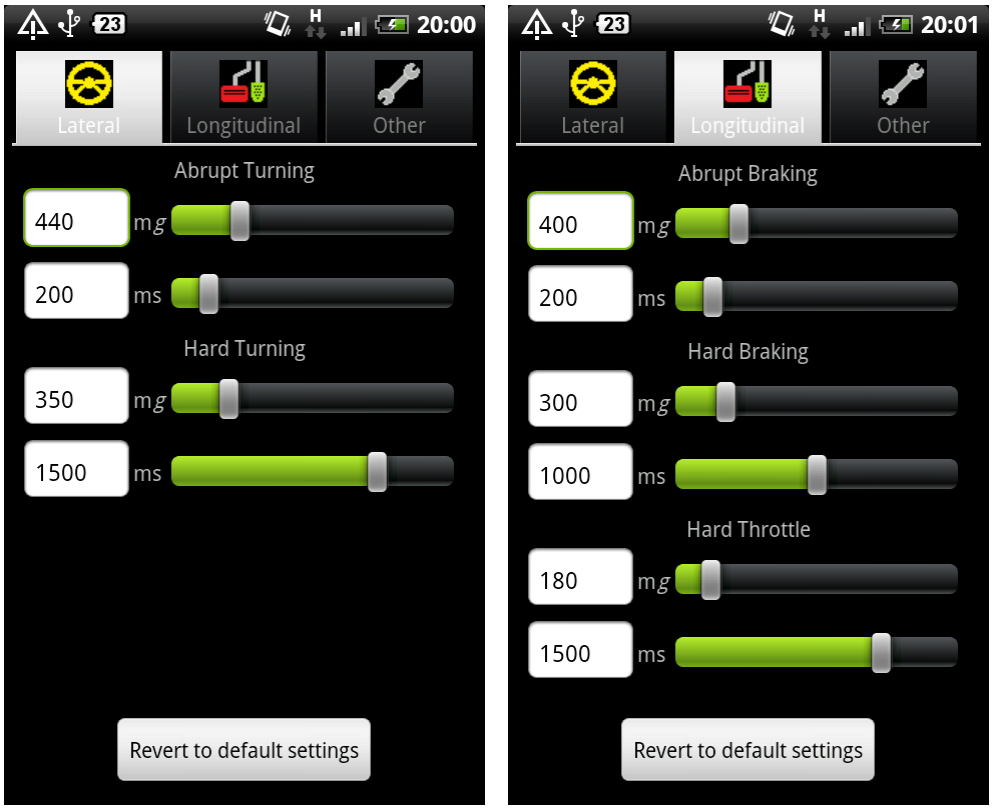


Figure 4: DriveMod settings activity showing tab for 'Lateral' event settings; DriveMod settings activity showing tab for 'Longitudinal' event settings

The app successfully records all the data it is meant to record—trip and event Id numbers, event types, date and time, location, accelerometer readings in R-F-U co-ordinates, and audio recordings (which are stored on the SD card in the DriveMod folder). The app also presents data the user in accordance with its design. The screenshots in Figure 5 demonstrate the data presentation capability. Additionally, the application functions with the screen turned off during a trip.

The one exception to complete functionality is that the calibration does not occur if the screen is turned off during the 20 second countdown at the beginning of a trip. On the other hand, if the screen is turned off after this countdown completes but before calibration has occurred, then calibration is still successful. While investigating this issue, we noticed that calibration is still successful in an identical scenario if the application is running through the debugger, even if no breakpoints are set. Without further investigation, our best assumption is that there is a timing issue that occurs

between starting the service (and by extension the accelerometer callback function) and the onPause event occurring for the previous activity, which does not occur if performance is degraded when using the debugger. A simple fix would be to include an instruction, in the app's countdown screen, for the user not to turn off the screen before putting the phone away. Most phones' screens time out in any case, but we could do even better and programmatically turn the screen off after the few seconds of calibration.

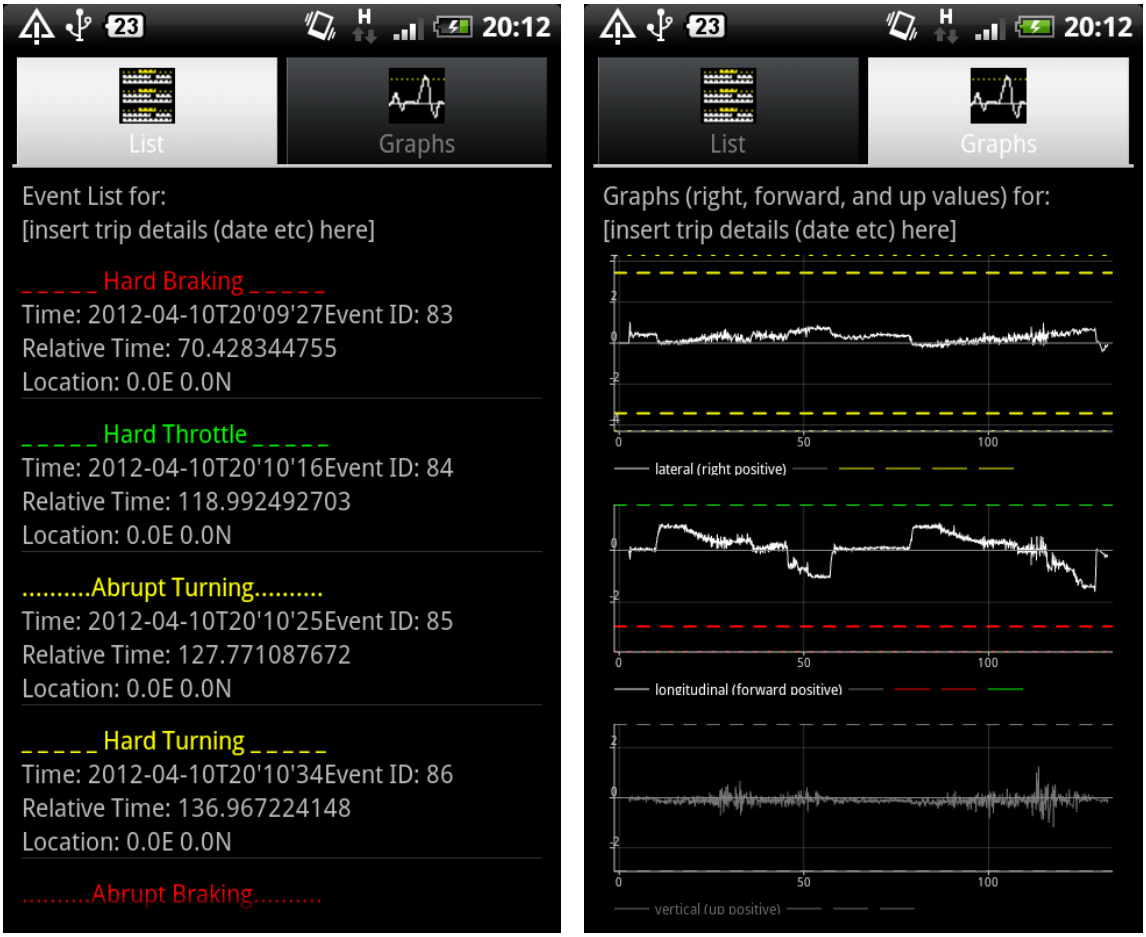


Figure 5: DriveMod previous trip data activity showing event list tab; DriveMod previous trip data activity showing R-F-U acceleration graph tab

4. Stumbling Blocks and Lessons

The single biggest problem the team encountered was that some Android phones do not comply with the 'hints' (this is Android terminology) for how often a programmer wishes a program to get accelerometer values. The entire time aspect of the event-detection was originally programmed assuming a known sampling interval, but this had to be completely recoded to explicitly check timing. The instability of the spacing interval seems understandable, but the issue of hints being ignored by some phones would seem to be an actual fault with the implicated phones

The team also should have done is processing of accelerometer values in a service from the beginning. It was always true that the app was supposed to run (during a trip) with the screen off, so we should not have waited to make that possible.

5. Organization and Division of Labour

Frances, a programmer on the team, was responsible for coding the fundamental app area of accelerometer reading, calibration, co-ordinate swapping, and event detection. In the middle stages of the project she did troubleshooting on the app running slowly, and helped uncover the problem of different devices responding differently to requests for specific accelerometer sampling rates. This device functionality flaw issue also meant she had to reprogram all of the event detection logic. She also created the activity for a trip in progress. This is an artefact of the programming process and not necessary to the final app, but it was required for the extensive testing required.

Matthew, the other programmer on the team, programmed the collecting of GPS data and microphone audio as well as the output of feedback audio messages. He also handled the database interactions, including recording all event and trip information. This included saving all of the values from the settings as well, and he made the textbox-slider slider combination settings adjusters work smoothly. In the late stages of the project he moved

the accelerometer code into a service, and he was instrumental in diagnosing many small problems with the code.

Adrian, the apper on the team, was responsible for all planning for the functionality of the app and all assignment of work for the project effort. He also liaised with several members of his department to validate specific elements of the design and ensure its pertinence to the field. He prepared the basic algorithms for calibration and event detection and helped research and develop the vector algebra for co-ordinate swapping. With the help of Matthew, he learned XML and did all of that part of the app so as to be able to do UI creation independently. He then began to learn programming, and he ended up completing many programmatic aspects of the UI as well, correcting some vector algebra code, and handling all of the graphing in the app. Toward the end of the project he also designed the tab and app icons and did real-world testing in order to determine suitable default settings for the events.

6. Apper Context

The short-term potential for DriveMod to generally advance performance in the task of driving is of course due to it providing feedback to the end user—any driver who goes to the trouble of finding and installing the app. At a minimum, the driver's awareness of when he or she may be driving poorly will be increased thanks to the audio feedback messages. This will hopefully encourage safer and more efficient driving. The more advanced end user may also benefit from perusing collected data while not on the road. He or she should at least notice how often certain events are detected, and whether efforts to decrease this over successive trips are successful. Some appreciation for the effects of the research-oriented factors recorded may also be discovered by such a user.

The major and longer-term contribution of DriveMod to the field of Human Factors Engineering is that it can give researchers relatively easy access to the types of data it records. Its ability to collect accelerometer data in particular is important, since steering corrections are known to be a good indicator of driver drowsiness[2] and "Delayed event detection and degraded vehicle control are observed when drivers fuel their need to perform extra-driving activities"[3]. 'Instrumented cars'—equipped with accelerometers and other extra sensors—are used in many experimental and naturalistic studies on driving, and an app that can be installed on phone and placed in a car obviously represents a less costly alternative to custom outfitting of a vehicle with dedicated sensors. The installation cost drops to zero when one considers how many people already own devices that could run the app. The picture improves still further with the prospect of an evolved version of DriveMod transmitting data back to researchers by e-mail, or any other method at a distance, and reducing not just the need for researchers to access the vehicles but the study participants as well.

It is worth mentioning here that acceleration data *is* actually being recorded on modern cars every day; unfortunately the system that does so is not easily interfaced with. The electronics that control airbag deployment are mandated to collect acceleration and other

vehicle data (such as control inputs and speedometer readings), but this is generally available only for accident reconstruction efforts, so again the problem is access.

With DriveMod recording raw vehicle-axis acceleration data, instances of detected events (adjustable in force and time span), location, and ambient sound, there are many possibilities for research. The correlation (or causation, in an experimental setting) between any combination of the five event types and any of the following factors could be studied, meaning:

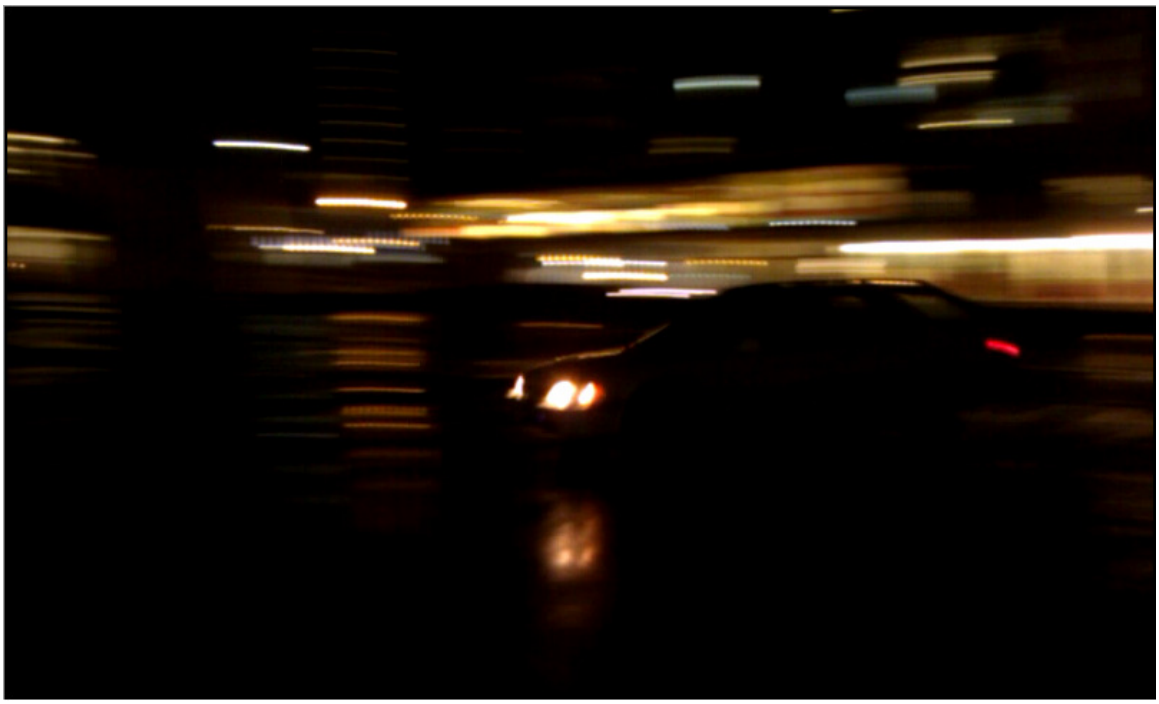
a) Driver circadian rhythm state (time)
b) Weather (time + location)
c) Traffic density (time + location)
d) Roadway design (location)
e) Conversation with passengers (audio)
f) In-vehicle entertainment system use (audio)
g) Communications device use (audio)

Given all of this, DriveMod should be able to find some place in helping human factors researchers, roadway designers, and drivers themselves.

7. Future Work

The development team would like to make the forward-detection routine work for rearward departures and even gentle departures from a vehicle's parking spot. This would be difficult but would be a major step toward having the app be universally deployable.

Though DriveMod was conceived primarily to serve the needs of the research world, the development team would be interested in exploring working with a business school to promote its use by non-academics.



References

[1] Transport Canada, "Canadian Motor Vehicle Traffic Collision Statistics: 2009" [Online] (Available: http://www.tc.gc.ca/eng/roadsafety/tp-tp3322-2009-1173.htm)

[2] "Fatigue of Car Drivers – Detection and Classification Based on the Experiments on Car Simulators", P. Bouchner et al., Proceedings of the 6th WSEAS International Conference on Simulation, Modelling and Optimization, Lisbon, Portugal, September 22-24, 2006

[3] "Behavioral Entropy as a Measure of Driving Performance", E.R. Boer, Driving Assessment 2001: The First International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design