A methodology for inexpensive GPS data storage and analysis

Tomas Levin
Transportation research
SINTEF technology and society
Trondheim, NORWAY
Tomas.Levin@sintef.no

Abstract— The increase in the use of tenders in Norwegian transportation research is forcing the research community to think in new ways. This paper looks into using open source software and consumer grade equipment for scientific purposes. The use of consumer grade data logger and open source hardware is a viable alternative. This paper documents the methods and the tools used for three research projects, 2 are close to finishing and one has just started.

Vehicle speed and positional data was collected with inexpensive data loggers and data was stored and analyzed with open source tools. The tools proved to be more powerful, but they do not offer the same graphical user interfaces. The main way to analyze data is through the use of standard SQL. Both average speed calculations using a road network ana driver behavior analysis can be carried out we few lines of code for large amounts of data.

GPS; data storage, open source; map matching (key words)

I. INTRODUCTION

In the development of transport models that have a supply side driving speeds are a key data to use for verification. From an emission perspective there has also been developed average speed emission function. The challenging part of the average speed emission functions is the fact that emissions have a near exponential growth in the lower speed region. Figure 1 shows an example of a EURO 4 34-40 ton truck fuel consumption. It is quite clear that if the wrong speed is used for emission calculations the error could be severe. To get as accurate emission estimates and good supply side transport models it is imperative to get good speed data for a city or region.

The traditional collection of speed data has been through the use of a few floating vehicle driven for registration purposes and manual calculation of driving speed have been conducted. In this paper we are suggesting to use GPS data collected from a pool of vehicles without specified routes to collect travel time data. Also due to the Norwegian Public Roads Administrations use of tenders for commissioned research one has to look at cost effective ways of collecting data and processing data. Thus a viable option for this project was to use open source software.

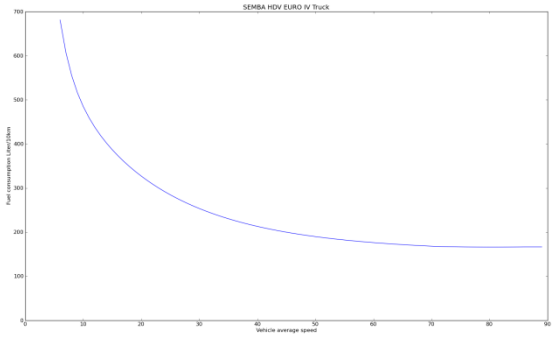

Figure 1 Truck 34-40 tons EURO IV fuel consumption

The methodology described in this paper has been used in 3 research projects: The Green Freight Transport Project, The speed model for commercial vehicles project and is to be used in the Green Activety Zones project.

II. DATA COLLECTION

Professional GPS tracking units for logging data have been on the market for quite a while. What is new is the amount of low cost consumer grade GPS units with data logging capabilities that are on the market. Several units where tested, the cheapest units use the MTK-II chipset with 4 MB of data storage. This allows for storage of about 200k data points. 200k data points gives you approximately 55 hours of logging time at 1Hz. Our plan was to leave loggers in vehicles for an extended time period, 2-3 months at the time. Thus we needed loggers that where capable of storing data in the gigabyte magnitude. This could be achieved by the use of external logging units such as a phone running the Windows Mobile platform.

Low cost non differential GPS units have been proven to give good accounts of speed(Keskin and Say, 2006, Witte and Wilson, 2004).

A. *Windows mobile prototype*

The Windows Mobile platform was chosen because an interpreter for the Python programming language has been developed and is freely available. Python's genuine advantage is the amount of libraries that are shipped with the interpreter. It only takes around 400 lines of code to create an user interface, GPS logging and a FTP (File Transport Protocol) application for collecting, storing and transferring data when in the vicinity of known wireless networks. The developed application also collected data from the vehicles ODB-II connector via a Bluetooth OBD connector based on the ELM 327 chip. The application could run on any Windows Mobile device of windows pocket pc greater then version 5. The only requirement would be that the unit had to have Wi-Fi and GPS. The GPS could be an external Bluetooth device or internal.

B. *First test of devices*

Two Windows Mobile devices where tested by a local truck operator. The problem turned out to be the interaction with the drivers. In order to get Bluetooth setup correctly the units had to be started in a specific sequence. And the startup sequence took some time, about one minute. Then the users had to click on the device to start the recording. The startup procedure turned out to be too much hassle for the drivers, so that they quickly stopped using the loggers. After talking with the truck operators it was agreed to try with black box units. A set of black box units where developed, they lacked the ability to transfer data when in vicinity of known Wi-Fi zones. The data-loggers stored data to 2 gigabyte SD-cards. Thus the SD card could quickly be extracted from the logger unit and replaced with a new empty card. The units were placed inside the vehicle with the antenna on top of the dashboard.

1) *Using standards*

To ensure compatibility between loggers and possible different logging platforms all speed and position data was stored as standard NMEA sentence. Only 2 sentences where needed, the GPRMC and the GPGGA sentences. The benefit of using the NMEA standard is that it is one that is supported bye most GPS units. And if the standard is not used NMEA sentences could be constructed from available data. Another reason for storing the data in the NMEA format is that many applications are able to read and display NMEA data.

Using the NMEA standard had the benefit of separating the logging from the analysis process. Basically any logging equipment could be used as long as it returned data in the NMEA format. NMEA is a proprietary standard is controlled by National Marine Electronics Association as sells for around $325. But most of the application level has been reverse engineered, a good source of documentation is found in the appendix of (UBLOX, 2006).

2) *Test of device data quality*

To test the data quality of the cheap GPS devices a vehicle that has a professional 100 Hz GPS unit was used as a reference. The cheep GPS units where placed inside the card while the 100Hz GPS had a roof mounted antenna. The effect of placing the GPS units inside the vehicles is that sky is not visible in the opposite direction of driving. Figure 2 shows the placement of the GPS units on the test vehicle. The expensive GPS unit is designated as VBOX, while the cheap GPS logger units where designated as RTCU.

Figure 2 Placement of antennas on test vehicle

A test route was created; the loop was run two times. Two types of test where conducted, dynamic test when the vehicle was in motion and a static test when the vehicle was parked in an urban setting. The analysis later reviled that the cheap GPS units had a data loss of 10%. This was later traced to a programming bug.

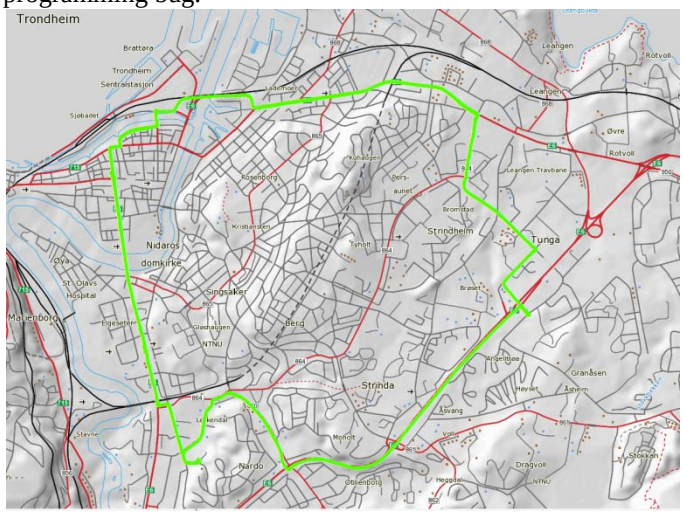

Figure 3 Map of GPS testroute (source: Statens Kartverk)

Table 1 Data points in each test

Test name	Data points		
	Vbox	RTCU #2	RTCU #6
D1 – dynamic test 1	1571	1407	1418
D2 – dynamic test 2	1656	1456	1480
S1 – static test 1	1201	1081	1079

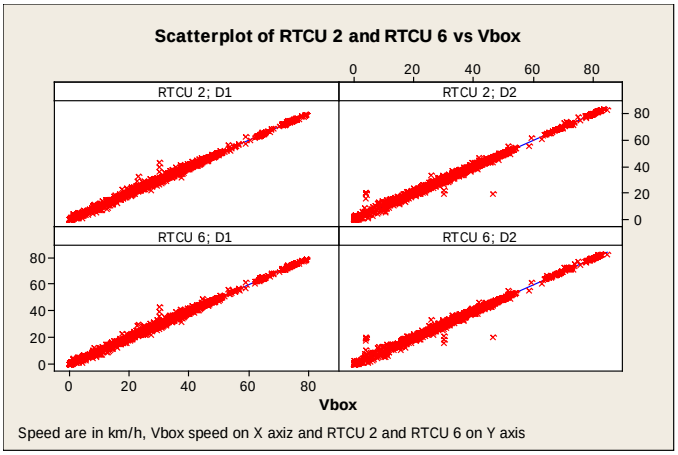

Figure 4 Scatter plott of GPS speed of VBOX versus RTCU units

A simple regression analysis gave the following results:

```
RTCU 2 = 0,261 + 0,994 Vbox
R-Sq = 99,5%

RTCU 6 = 0,106 + 0,997 Vbox
R-Sq = 99,4%
```

There are some large discrepancies in Figure 3, these discrepancies where found under a railway bridge in dynamic test 2. It seems that the cheaper units have some sort of smoothing algorithms that filtered out erroneous data under the bridge. The more expensive unit seemed to report unfiltered data.

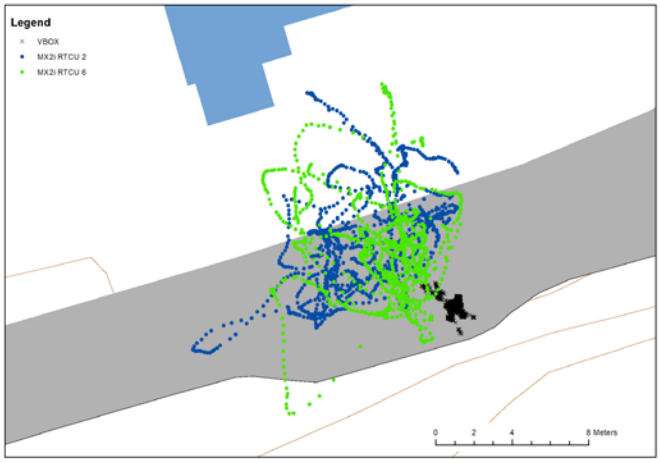

Figure 5 Static test of GPS positional stability

A static test was conducted to check positional quality in an unban setting, in this test the view to the north east is blocked by buildings and the vehicle is facing the west. Her we can clearly see that the more expensive unit that has an antenna placed on the roof is more stable (the black marks).While the two cheaper units with the antennas placed inside the vehicle have a lot more positional drift.

From this simple test is was believed that the data collected from the cheap GPS units was accurate enough for our analysis of average speed on short road links. Average load link length was about 200 meters.

III. SOFTWARE FOR STORING GPS DATA

A total of 10 data loggers where commissioned and these would log data from truck running in semi fixed routes. We expected a data volume in excess of 10 million data points. It turned out that we recorded nearly 6 gigabytes of data from these trucks. This was a bit less than expected and was due to the fact that some drivers unplugged the GPS units for reasons unknown to us.

Our plan for analysis of the data relied on the use of GIS techniques, map matching and linear referencing thus we needed to store the data in a georeferenced format. For analysis after the GIS operations we needed database capabilities and data export functionality. The PostgreSQL database was chosen for storing the GPS data. This was due to the fact that it is open source, it can handle extremely large data tables(32 terrabyte) and finaly it has a opensource GIS extension called PostGIS. The PostgreSQLdatabase runs on several operating systems including the windows operating system. But due to the large amount of data we expected a 64 bit FreeBSD server with 8 GB of memory was chosen. The advantage of this was that we could use a 64 bit version of PostgreSQL that could utilize more than 4GB of memory. The hardware was a AMD x4 Phenom processor running on 3.4 MHz, an el-cheapo motherboard stuck in a rack case with 2 mirrored 10k rpm disks for database storage. The cost of the hardware was about €1000. The FreeBSD operating system is an open source UNIX variety available free of charge. The benefit of a FreeBSD system compared to a Linux system is that the kernel, user land and and applications are managed by the FreeBSD development team. This should theoretically give a more stable system. Current uptime of system is approaching 115 days. The server has not been rebooted since installation.

Both FreeBSD and PostgreSQL have been around for a long time and have developed a quite large user community. In praxis this means that you will quickly find solutions so problems encountered simply by Googling error codes or more general issues. The PostGIS spatial extension to the PostgreSQL is a relative newcomer to the scene. PostGIS is developed and maintained by a Canadian company called Refraction and is also available as open source and free of charge. The software package is built around open standards and has quite good documentation.

A. Duality of GIS

When speaking of GIS people first and for most think of desktop GIS applications that can create nice maps and graphically analyze data. But users can also interact with the GIS systems at programmatic level. Suppliers of GIS tool have for a long time given users the possibility to access, manage and manipulate data through custom API's (Application Programming Interface). Thus if you are a proficient programmer you should be able to use both modes.

PostGIS does not offer an API in the classical sense, it rather integrates with the database and becomes an extension to the SQL database language for relational databases. The effect of this is that any supported GIS operation can be used trough the use of standard SQL statements. The combo of PostgreSQL and PostGIS does not offer a graphical interface. For this the user has to use a desktop GIS client. There are some graphical clients on the market that offer direct connection to the PostgreSQL database such as UDig and QGIS. ESRI's desktop systems can also connect to the PostGIS database if one buys an substantial database extension, ARCSDE, or if third party software like zigGIS. For our projects Qgis was used for graphical inspection and digitization of new features while all the rest of the work was done from the command-line in SQL.

The setup of our system makes little usage of the duality of GIS and focuses heavily on moving tasks that used to be done graphically over to the SQL language. The benefit of this is that the user will create a script file that is the recipie of the whole GIS operation. If new data is added it is only a question of running the script to create new result data. The script for importing 10 million + data records and matching them to the road network, error control of data and grouping to a suitable export table is 894 lines long including code to create the database tables and comments. Running the script on the server described earlier takes 6 hours to complete. On the ESRI platform and with a 3GHz computer with 4GB of memory and windows XP it took 5½ day to complete just the map matching. The extreme difference in execution speed is believed to be linked the fact that ESRI has very general interfaces to functions while in PostGIS the user uses just the parts needed.

B. Writing specialized functions

To extract and transform the GPS NMEA files into database tables and split files into journeys. A journey is defined as a sequence of data points where the speed is over 3 km/h, but if the speed drops under 3km/h for s short period (180 seconds) the data sequence is not split into different trips. The PostgreSQL database allows for incorporating different programming languages when writing creating functions. Functions in PostgreSQL are created with the use of SQL, thus the conversion code that was written in Python for the ESRI Geoprocessor could be included by simply removing the Geoprocessor specific commands. The code sample below shows the famous Hello World program written and executed as SQL.

```
CREATE FUNCTION HelloWorld ()
  RETURNS text
AS $$
  # PL/Python function body
    return "Hello World"
$$ LANGUAGE plpythonu;
SELECT HelloWorld ();
```

PostgreSQL's ability to include other programming languages meant that we could reuse code that was written for the ESRI Geoprocessor with very little porting effort. It also reduced the amount of programming languages the user needs to be proficient in. It is sufficient to know Python and SQL.

The final point with PostgreSQL is that it comes with standard ODBC driver for Windows. Thus data tables linked to Windows applications with ease. In our case results from the analysis was imported into statistical analysis software like Minitab and SPSS.

IV. MAPMATCHING AND MITIGATION OF ERRORS

There exist quite a few map matching algorithms for real time mapping of GPS positions(Quddus et al., 2007). Some of these are quite complex and when map or positional accuracy algorithms can get quite complex(White et al., 2000). In this paper we will be focusing on developing methodology for mapping GPS data to the road work in a simple, but powerful fashion. The idea is not to remove the errors in the map matching process, but to find ways to mitigate erroneous matchings.

A key point in our setup of the system is that it should be able to collect data in a loose fashion without definition of specific routes. The map matching routine is extremely simple, it is based on distance to closest road and the GPS data is snapped to this road. This approach will give erroneous snapping of points to roads nearby, especially at intersections.

It is here were we apply the mitigation approach. The parameter that we are looking for is the average speed on the link. The average could be expressed as an average of the instantaneous speed observations or as distance traveled divided by time used.

$$\hat{V} = \frac{\sum v}{n} \qquad \text{Equation 1}$$

$$\hat{V} = \frac{distance}{time} \qquad \text{Equation 2}$$

Equation 1 is more sensitive to errors in map matching then Equation 2. By using the second equation intermediate points on a road link are of less importance, time and distance between the entry and exit point on the link are the important factors. As seen from the registration test data static positioning is likely to lead to increased erroneous matching at intersections where the vehicle is traveling at low speed or standing still. The GPS units also seldom report a speed of 0 when standing still, this could lead to a higher than real average speed. But the real selling point for using Equation 2 is that the extra data needed for the speed calculation, distance traveled, can be used for mitigation. Figure 5 shows a sketch of the map matching routine. The map matching algorithm is as follows:

1. Split GPS data into trips, a trip is a sequence of observations with speeds over 3km/h including shorts stops under 180 seconds.
2. Find closest road link to GPS point.
3. Calculate linear reference of GPS point.
4. Calculate time and road link distance between first and last GPS points on road link for each trip.
5. Calculate GPS vector and road link vector to find direction of travel, along or against digitized line direction.
6. Percentage of distance traveled on road link, the distance between the first and last GPS point and calculate percentage of road link.

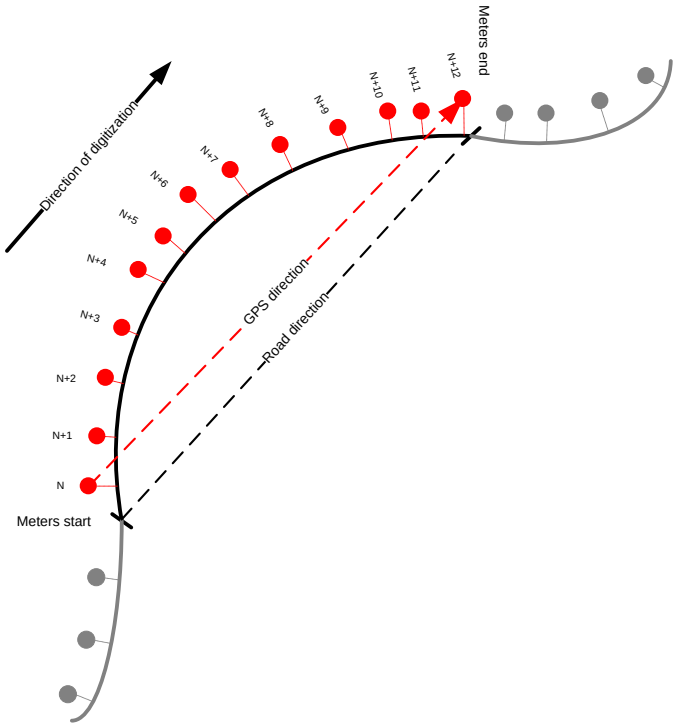

Figure 6 Sketch of map matching

A. Loose data collection scheme

The idea behind having a loose data collection scheme is to collect large amounts of data and drop data that is believed to be erroneous. The map matching algorithm generates a table of distance used for average speed calculation and distance of road link. Data are dropped if the average speed is calculated for less than 75% of the link length. There is still a problem if the link length is very short. In this project the standard road segmentation was used. Thus we opted to remove links that were less than 200 meters as we were primarily interested in long-haul trucking operations.

One challenge in urban areas is to record driving speeds on different road links or longer road segments. The power of the map matching algorithm is that it runs purely in SQL on the server side. The map matching routine is very simple and robust, it is not as advanced as the routines described in (Quddus et al., 2007). But for long-haul operations the routines seem to give quite good results, of a set of 8,4 million points 7,8 millions where given road a road reference, direction of travel and road gradient. For the Green Freight dataset average speed was calculated based on 3,0 million observations left. The mitigation process removed 117651 erroneous data points, the rest were removed because we only wanted to study roads typical to long-haul rural trucking. If the trips in urban areas have been included this number would probably have been much higher.

Table 2 and Figure 6 show results form an analysis of driving speeds for long-haul trucking operations in rural areas compared with the speed limit.

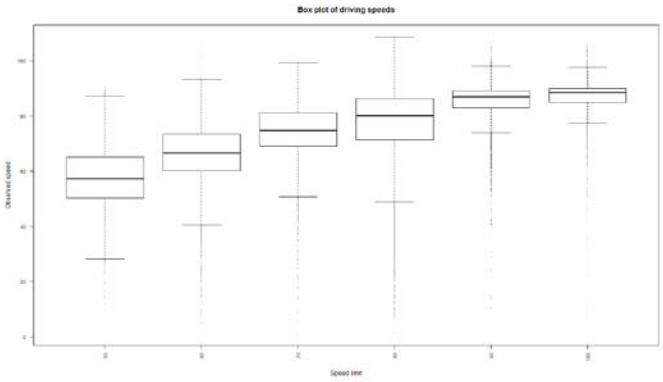

Figure 7 Box plot of driving speeds for long-haul trucking operations in rural areas

Table 2 Driving speeds for long-haul trucking operations in rural areas

Speed limit	Mean observed speed	Standard Deviation	95% CI
50	57,8	11,9	57,4 - 58,2
60	66,4	11,2	66,2 – 66,6
70	74,6	9,3	74.5 – 74,8
80	77,4	11,4	77,3 – 77,5
90	85,1	6,48	85,0 – 85,2
100	86,6	8,49	86.3 – 86,9

The statistical analysis was performed on a windows pc and the data was accessed through a standard ODBC database driver and processed in a standard statistical package, in this case R[1].

B. Further analysis of driving behavior

Unlike spreadsheets the database has no notation of the rows before or after the row that it is currently processing. But in the ISO SQL:2003 standard a window function appears. In laymen's terms this feature allows the database to access rows before or after the current row. PostgreSQL is the only open source database to include this feature at present. The window

[1] http://www.r-project.org/

function appeared in version 8.4 of PostgreSQL. The window function really shines when it comes to looking at driver behavior.

A simplistic way of calculating accelerations form GPS data is to divide the speed difference between two points by the time between the two points. As part of the Green Freight project one wanted to look at driving behavior in relation emissions. One way to do this is to look at RPA which is believed to be correlated to driver emissions(Ericsson, 2001). RPA is defined in Equation 1. T is the length of the micro cycle, v_i is the instantaneous speed, $a+_i$ is the instantaneous positive acceleration and x is the distance of the micro cycle.

$$RPA = \frac{\int_0^T (v_i * a_i^+) dt}{x} \qquad \text{Equation 3}$$

Using the window function in PostgreSQL made it easy to generate RPA measures for every micro cycle found in the GPS data. A total of 15258 driving cycles where found in our data. This data was then compared to the RPA of the micro cycles found in the ARTEMIS project.

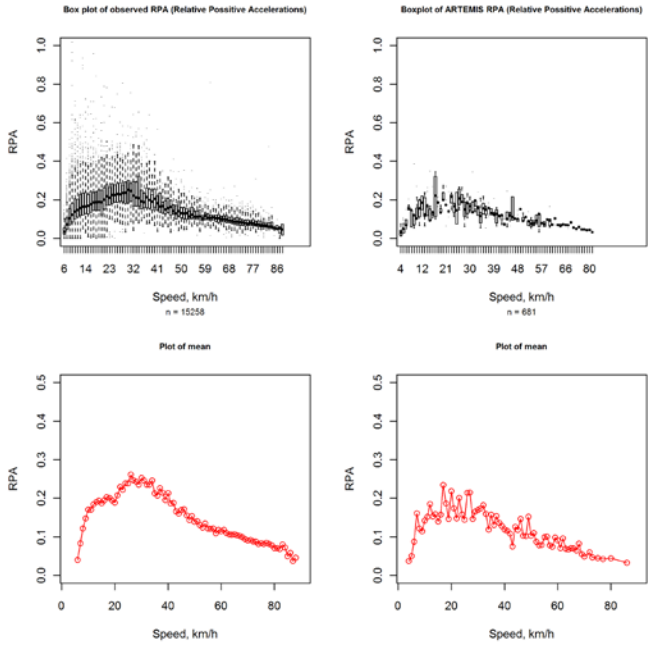

Figure 8 Comparison of RPA for heavyduty vehicles, observed and ARTEMIS

Figure 7 shows a comparison of the RPA for the long-haul heavy duty vehicles we observed and the RPA for heavy duty vehicles found in the ARTEMIS application(Keller et al., 2007). Each micro cycle was grouped by its average integral speed. The RPA analysis seems to indicate that RPA found in observed micro cycles is comparable in terms of scale and change as a function of average speed.

Two SQL statements of 35 and 23 lines were required to produce a table with micro cycles, average speed RPA and percent of time accelerating, decelerating and cruising. The database was created with the same SQL code as for the speed studies.

V. GPS ARTIFACTS

When looking at a speed/acceleration plot for the whole dataset we discovered something that could best be described as a GPS artifact. Figure 8 shows a dot plot of speed and acceleration. There are extremely few points with low speed and high accelerations, but with over 7 million data points there should probably have been a few faulty registrations a low speeds, especially since this plot includes driving in urban areas. There seems to be a line where data is cut off, it is a quite clear straight line. Looking at the documentation for the cheap GPS receivers give few clues to why this is happening. But the fact sheet of the GPS chipset gave us a clue; the GPS data is post processed before sent to the user. And according to the fact sheet: "TheANTARIS4GPSengine inside offers outstanding navigation performance in the most challengingmetropolitan areas(UBLOX, 2006)" The GPS unit is also said to run at 4Hz, but only reported data at 1Hz. It is not mentioned if the unit averages 4Hz to 1Hz to get better positional estimates.

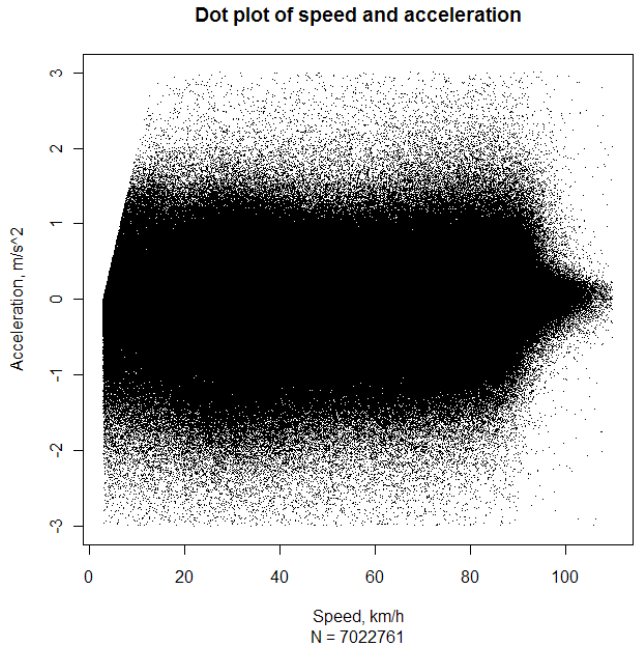

Figure 9 GPS speed and acceleration artifact

This discovery sparked of a test of other GPS chipsets, mainly a cheap receiver with the MTK II (Media Tek Inc) chipset. The Holux M-1000 that was used in the initial prototyping runs at 1Hz, but again the chipset documentation indicated 5Hz. The update rate of the Holux was successfully changed to 5Hz with the use of a small application from Media Tek Inc called GPS Mini[2] written. The track from a device running at

[2] www.sparkfun.com/datasheets/GPS/MiniGPS_1.32.zip

1Hz seems more stable then the track from a 5Hz model. To us this indicated that when external reporting is set to 1Hz there could some sort of averaging going on within the unit. No further testing at 5Hz, but we had found indications that 5Hz could be achieved with inexpensive units.

VI. DISCUSSION

As a response to the Norwegian Public Roads Administration to cut research cost by the wide spread use of tenders for research project on is forced to look at ways of cutting cost. One way to do this is to move away from costly registration schemes and use of expensive software. This paper has looked at inexpensive ways of collecting data from long-haul trucking by the use of inexpensive GPS units and open source software for storage and analysis.

Using cheap over the counter GPS-logging units seems to give good results. As the miniature test of the inexpensive GPS units compared to the expensive professional units show quite comparable results. But the tests also proved that there was a need for a professional unit to compare the units with. The large scale registrations could be done with the cheap unit, but there was a need for a professional unit to benchmark against.

When using inexpensive GPS units its important to look at who is manufacturing the GPS chips. More data is often found from the chip producers then the manufacturers of the whole unit. The GPS units may not report data at the same rate as data is used internally. This fact opens up the possibility for the chip or unit producers to average positional data to give more accurate positional fixes. There can also be special functions hidden in the units that make them more or less suitable for different logging purposes. The documentation of the inexpensive over the counter units that we have used did not provide all the answers that on needs before conducting experiments. Thus if one wants to use inexpensive GPS units piloting is highly recommended. That is one way to counter the lack of documentation desired for scientific experiments.

We found open source solutions for storing and analyzing the collected GPS data. There are a few alternatives out there, but the PostgreSQL and PostGIS combo turned out to be both powerful and have the needed features. PostGIS and PostgreSQL outperformed commercial software like ESRI's ArcView platform when it came to linking observations with roads. It should be noted that the PostGIS and PostgreSQL has a steeper learning curve then the ESRI products due to good graphical interfaces. Performance wise the PostGIS and PostgreSQL solutions out performs the standard ESRI products. Linking GPS observations to road segments takes 10 minutes when run in the PostgreSQL and PostGIS environment. While the same matching routine takes 5 days in ESRIS's ArcMap application.

The command line approach has a high learning cost and initial cost for the first analysis. But for subsequent analysis the same could be reused with little or no modification. This is a clear advantage for the PostgreSQL and PostGIS applications. But it should be mentioned that ESRI also has an intuitive scripting environment (model builder) that could be used for repetitive tasks.

Robustness and simplicity is where the PostgreSQL and PostGIS suite excels. Procedures are written and executed as normal SLQ statements. Existing functions for data manipulation can be included in the database as SQL functions with little modification as long as the language is supported by PostgreSQL. Python, Perl and TCL are currently supported. This allowed for reuse of code written for the ESRI GeoProcessor.

A simple map matching algorithm was created and used quite successfully for long-haul trucking operations. But this method has its limitation when the road network gets more complex and GPS positional fix quality deteriorates. Thus it is believed that for urban operations an updated version of the map matching is needed. It would be beneficial to use an algorithm that is topologically aware and is run after the initial matching. A table of legal adjacent road links could be cheeked against previous and future positions. The key to this is the window function that is available in PostgreSQL. The algorithm cannot be used for real-time operations, but will work for historical data. Further work on this algorithm will be conducted in the Green Activity Zones project.

The speed data collected gives some concern of truck driving speed in areas with speed limits of 50 and 60. If one is to trust this data one has to be sure that the speed limit data is correctly entered into the national road databank. Most of the trucking operations were conducted during night time due to freight schedules in the companies involved.

The database setup is not limited to map matching or calculation of average speeds. Driving behavior based on the same GPS data is possible. The calculation of RPA figures for micro cycles was successfully conducted by adding only 58 lines of SQL code. Having data stored in the database and using SQL with window functions removes some of the need for purpose written computer programs. The code could be executed on the server side and exported to post-processing applications like statistical software through the use standardized database drivers like ODBC.

The use of open source is not without pitfalls. How do you know that PostgreSQL will be around in the future? The short and probably the safest answer is that you don't. But one key aspect of open source project is the use of open standards. If PostgreSQL and PostGIS stop development you will still have the possibility to export your data to other systems due to the use of open standards. Another issue is support, what do you do when something goes wrong or you just need help? This is where open source has its commercial side. There are companies built around providing support for the products. But user forums on the internet are also good sources of information and solutions to common problems. From a

researchers perspective trawling the online communities and online user commented documentation has been sufficient. For the moment there seems to be more effort put into getting lots powerful functionality into the programs then to make good point and click interfaces.

VII. CONCLUSION

It is possible to use inexpensive GPS units, but there care should be taken. The documentation of the devices is not up to scientific standards and therefore piloting becomes essential. The key is not to find the best unit, but the unit can deliver the quality needed. A reference system is needed to access quality under operating conditions. It is therefore essential to have access to professional quality equipment which has a known and clearly stated quality measure.

The open source arena has given rise to several applications that has the same functionality as existing commercial software. But the learning curve could be quite different; in general user interface and user interaction has been more developed in the commercial software. Open sources addiction to the command line is on one hand challenging for the user, but it is also excellent for repetitive tasks as you have a log of what the user have done. This log can then be run again for similar problems.

In our case the open source solution proved to be more efficient then its commercial desktop opponent. Thus data could be analyzed quicker and could be exported to other applications through standardized ways as through the use of ODBC.

The map matching routine developed for loose registrations works well and data can efficiently be stored and analyzed for rural operations where the networks are simple. For urban operations that map matching should be developed further to take into account road network topology, but still within the confines of standard SQL statements. This is believed to be achievable through the use of the window function so that calculations can use for past and future positions.

ACKNOWLEDGMENT

Many thanks to the NPRA for funding my PHD and a to the research counsil of Norway for supporting project like: Grønn Godstransport, Næringslivets fartsmodell and last but not least the Green Activety Zone project. We would also like to thank the freight companies who allowed us to track their trucks.

References:

ERICSSON, E. 2001. Independent driving pattern factors and their influence on fuel-use and exhaust emission factors. *Transportation Research Part D: Transport and Environment,* 6**,** 325-345.

KELLER, M., KLJUN, N., ZBINDEN, R. & WEG, M. V. D. 2007. Artemis / COST 346 - Road Model Beta-0.4d. Beta-0.4d ed. Bern.

KESKIN, M. & SAY, S. M. 2006. Feasibility of low-cost GPS receivers for ground speed measurement. *Computers and Electronics in Agriculture,* 54**,** 36-43.

QUDDUS, M. A., OCHIENG, W. Y. & NOLAND, R. B. 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies,* 15**,** 312-328.

UBLOX 2006. LEA-4A ANTARIS® 4 ROM-Based GPS Module Automotive Applications.

WHITE, C. E., BERNSTEIN, D. & KORNHAUSER, A. L. 2000. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies,* 8**,** 91-108.

WITTE, T. H. & WILSON, A. M. 2004. Accuracy of non-differential GPS for the determination of speed over ground. *Journal of Biomechanics,* 37**,** 1891-1898.