Aerospace Sensor Suite

ECE 1778 – Creative Applications for Mobile Devices

Final Report prepared for Dr. Jonathon Rose

April 12th 2011

Word count: 2351 + 490 (Apper Context)

Jin Hyouk (Paul) Choi: 998495640

Matthew Leigh Leonard: 992481202

Vincent Tarantini: 998328938

Contents

List of Figures

1 Introduction

1.1 Application Description

The Aerospace Sensor Suite (the application) is a mobile device application which allows users to **record** various data, **visualize** that data immediately on their device, and easily **export** the data for use on a pc. It is targeted to students, designers, and researchers in the aerospace industry as well as to radio-controlled (RC) vehicle and hobby rocket enthusiasts. It offers users the ability to transform their mobile device into a cheap, accessible, and easy-to-use orientation and position sensor.

1.2 Motivation

Obtaining information about a vehicle's position and orientation is a common practice in the aerospace research field. Specifically, students, researchers, and industry professionals in the state estimation, attitude and trajectory control and vehicle design disciplines need to test their designs on real vehicles. Purchasing and installing traditional sensors can be expensive and time consuming. The Aerospace Sensor Suite can be used to replace several of the most common sensors: accelerometer, orientation, and GPS; and, since it operates from a mobile device, is a much lower cost and hassle than dealing with multiple sensors with individual tasks.

Although there is definitely an argument that the quality of the data that can be achieved from the sensors on a mobile device is inferior to high-end aerospace sensors, it is believed that this application can still provide useful information and at a fraction of the cost and complexity than dealing with multiple, high-end sensors. The Aerospace Sensor Suite is only a starting point and is limited by the types and quality of current mobile device sensors. There is already a trend in mobile devices where each new generation has greater functionality, higher processing power, and more types of sensors with better accuracy. The 'smartphones' of tomorrow will increase the applicability and performance of applications of this type. Hopefully, the phone you carry in your pocket, in addition to the plethora of other uses, will also be the platform for a viable and trusted Aerospace Sensor Suite.

2 Overall Design

The top-level flow diagram is shown in Figure 1. It shows the five main screens: splash, sensor view, recording, saved data, and visualization. The diagram captures the application's main functionality. A detailed flow diagram, showing all twelve possible screens, is shown in Figure 2.

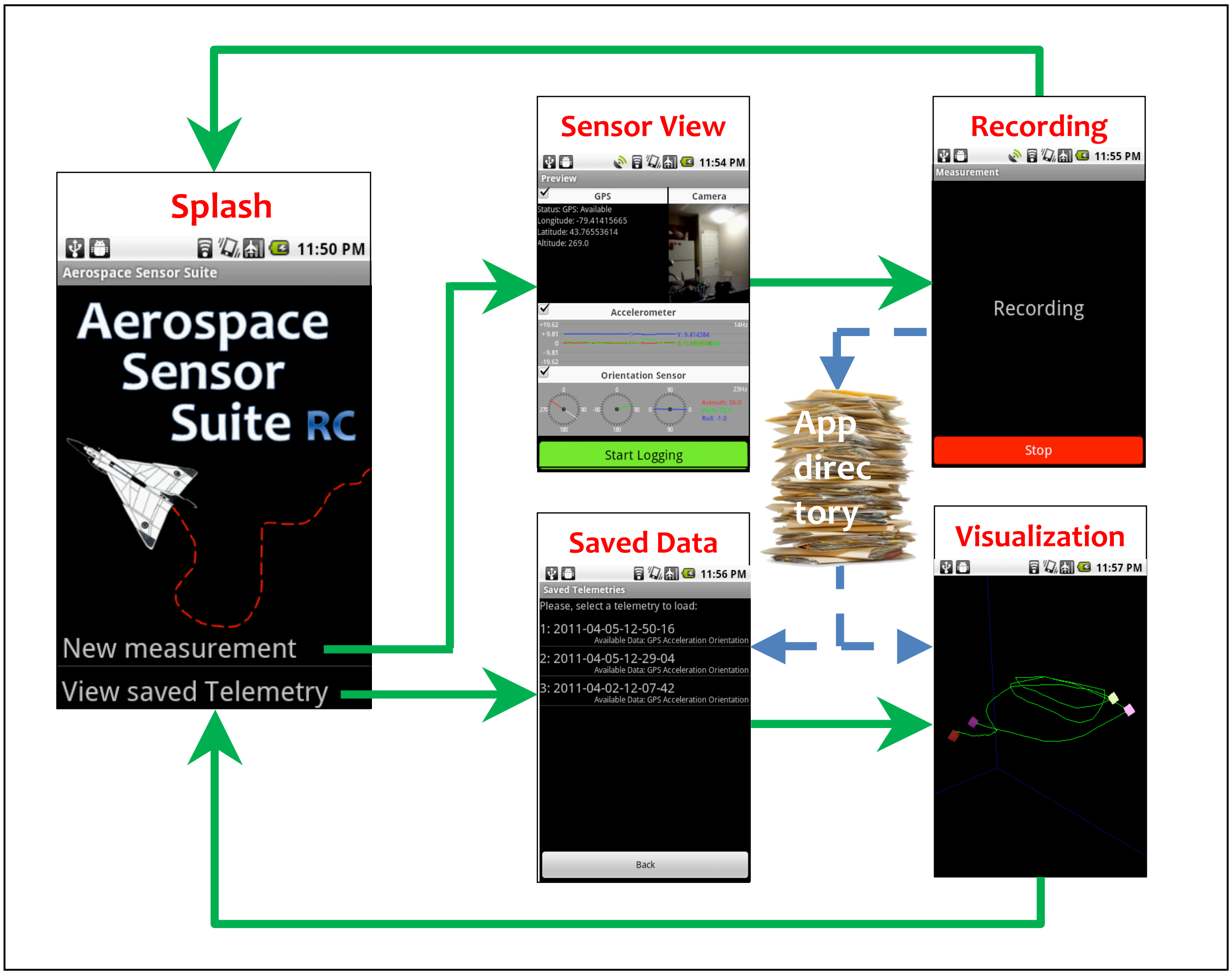


Figure 1: Top Level Flow Diagram

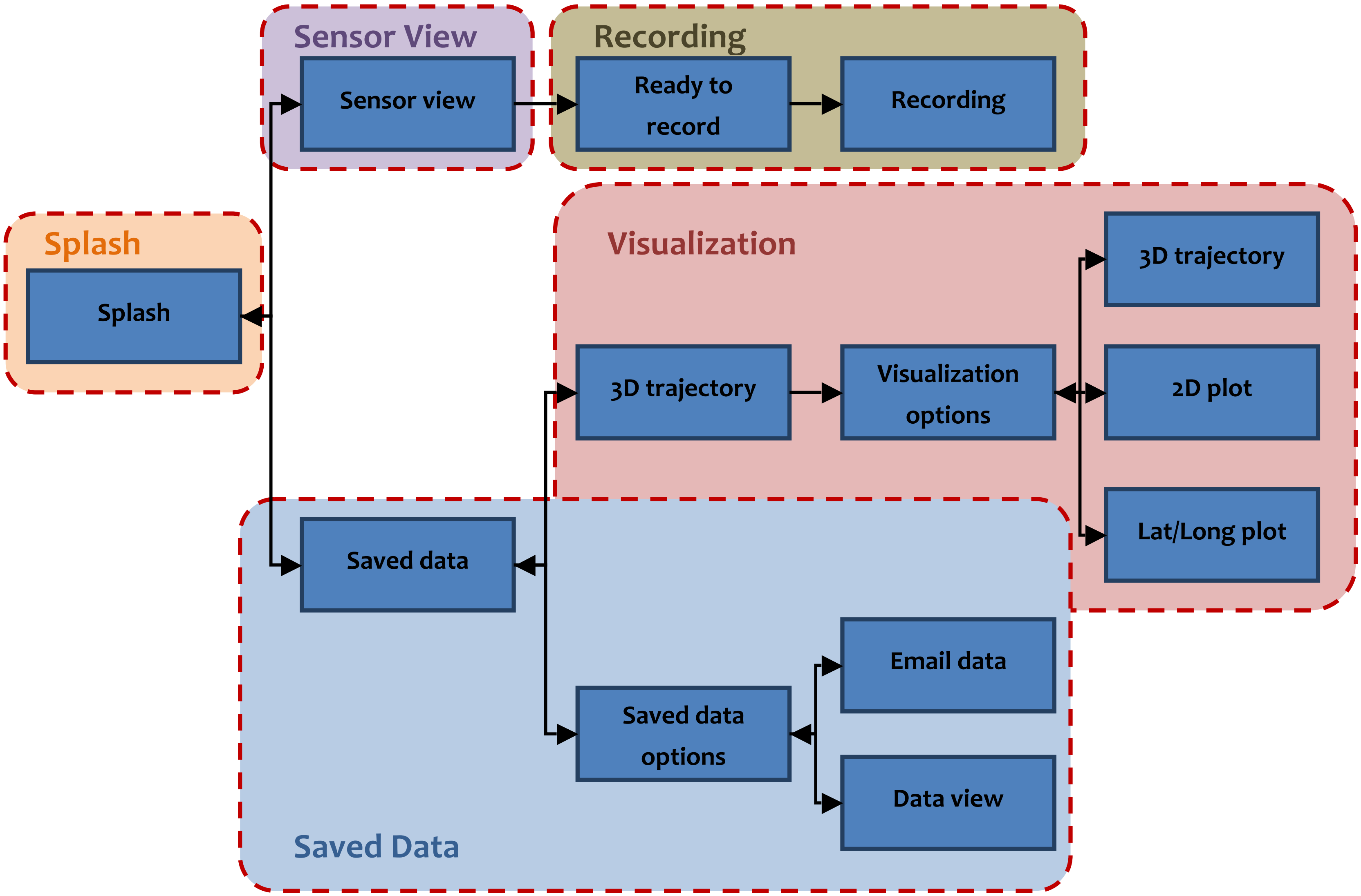


Figure 2: Detailed Screen Flow Diagram

3 Description of Screens

The screens shots and description of the blocks in Figure 2 are shown in Tables 1 and 2.

Table 1: Application screenshot images and description

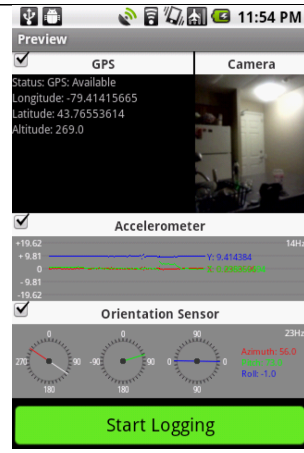
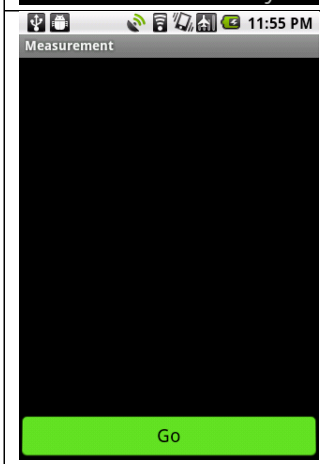
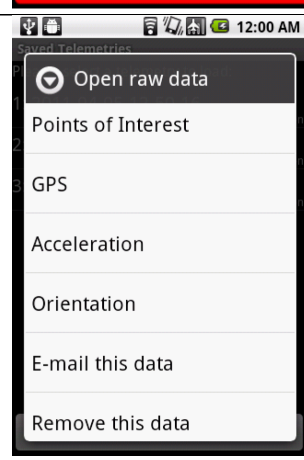
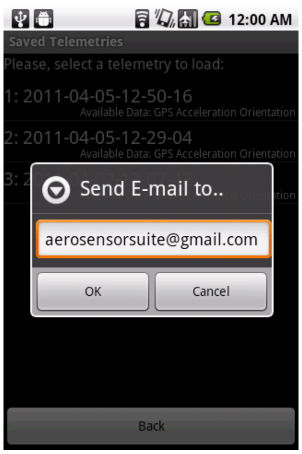
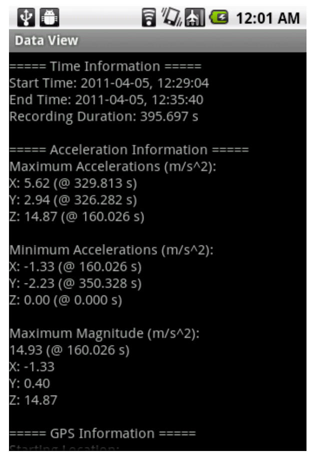
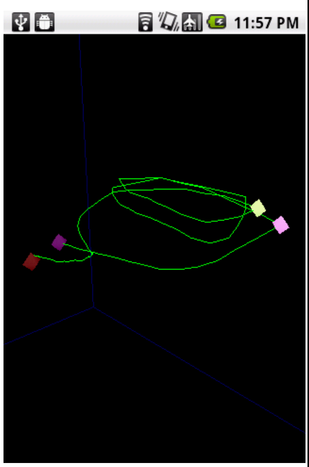
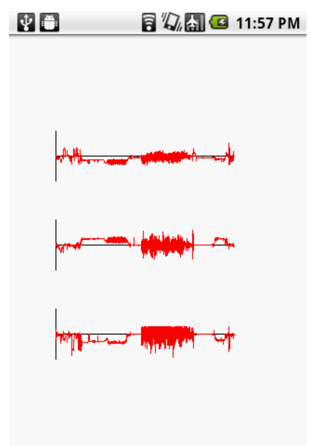
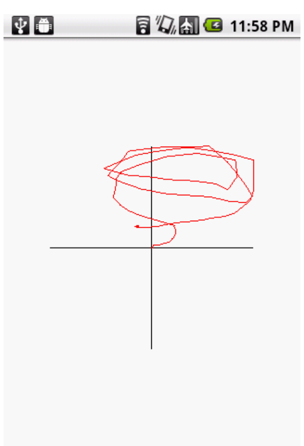
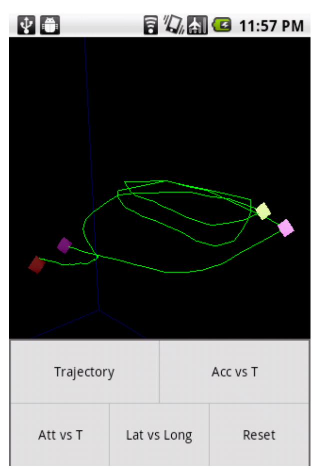
	**Splash** The splash screen is the main entry point for the user. The user may choose to start a new data recording or view previously saved data.		**Sensor View** The sensor view screen allows the user to preview of the raw sensor data – GPS, acceleration, and orientation – before the actual recording begins. The user may also select which sensors to record data from by toggling the checkboxes.
	**Ready to Record** When this screen appears, the user should mount the device to the vehicle. Once the device is mounted and everything is ready to go, the user can start the recording by clicking the "Go" button		**Recording** This screen appears once the user initiates the recording. It is simply letting the user know that it is recording. The recording process can be terminated by clicking the "Stop" button.
	**Saved Data** Allows the user to view previously recorded sensor data. The user can simply click the item in the list to view the visualized telemetry, or can click and hold the item to see more options with the selected data set.		**Saved data options** This menu opens when the user presses and holds a saved telemetry file in the Saved Data screen. There are several options including viewing the raw data and Points of Interest file, as well as the ability to email and remove the saved data.

Table 2: Application screenshot images and description, cont'd

	**Email** The Email screen allows the users to enter the recipient email address for the previously selected saved telemetry. After the user click "OK", the email is sent. Each data file is attached to the email in a separate *.txt file.		**Data View** In this screen the following items are displayed depending on the selection made by the user in the saved data options screen: raw GPS, accelerometer, and orientation data, and Points of Interest (information such as max altitude and maximum displacement).
	**3D trajectory** This screen presents the GPS data in a 3D latitude-longitude-altitude plot. The latitude and longitude have been converted from angles to ground distance. This view supports 3D manipulation via finger gestures. Maximum altitude and displacement are indicated with coloured markers.		**2D plot** The 2D plot screen allows the user to view a graph of X, Y, Z component of the previously recorded acceleration or orientation data with respect to the time. The user is also able to translate and zoom using intuitive finger gestures.
	**Lat/Long plot** This screen shows the latitude Vs. longitude graph of the recorded GPS data. The user is able to pan and zoom the view using intuitive finger gestures.		**Visualization options** The visualization options screen appears when the menu button is pressed from any visualization screen. The menu allows the user to toggle between visualization screens. The view of the current plot can also be reset to the initial state by clicking reset button in the menu.

4 In Retrospect

4.1 OpenGL

Over the course of this project the programmers learned quite a few lessons. The first is that OpenGL ES does not provide built in support for writing text to the screen. This is very different from their previous 3D programming experiences in DirectX, which does have direct text functions. When trying to incorporate labels into all aspects of the visualization, a task assumed to be trivial, it was quickly learned that this was not an easy task. In fact, many other developers ran into similar problems. After discovering this, the group was faced with a decision: try and get labels working or move forward with the rest of the visualization; it seemed obvious to choose to abandon the labels.

Another major lesson that was learned was that the Android framework and the OpenGL ES framework don't always play nicely together. We had hoped to include some Android controls on the OpenGL surface, but found the integration to be an extremely complex task, and, again due to time constraints were abandoned as well.

Finally, when trying to implement our own custom gestures, we ran into more OpenGL vs. Android problems where messages would be consumed by either Android, or OpenGL, and not being shared between both frameworks. If we were to do this project again we would probably try and find third party libraries for UI gestures, and for OpenGL text rendering so that we can focus on the main goals without having to reinvent the wheel.

4.2 State Estimation

One of the biggest challenges in developing this application was attempting to implement an estimation algorithm. The Apper spent a lot of time working with the recorded data trying to implement a filter that would improve the prediction of the vehicles. The process is referred to as state estimation and is commonly used in the aerospace and is an active research field. The concept is to integrate various forms of sensor data to achieve a better estimate of a vehicle's state. The state is usually the position and/or the orientation of the vehicle. For this application, the state was chosen to be the vehicle's position and orientation in three dimensional space. The sensor data used for this were position (from GPS), acceleration (from the accelerometer) and orientation (from the magnetometer). The goal was to gain a better estimate of the vehicle's state than the sensors could give directly. The GPS sensor is only accurate to about 5 to 10 metres and provides a measurement every 1 second, improving this accuracy and filling in position data in between GPS measurements would be part of the better estimation. This would be achieved by implementing either a filter or an optimization algorithm. In both cases the concept is essentially the same: the accelerometer data, combined with the orientation data is integrated to achieve a predicted trajectory (state vs. time). The prediction is then compared with the measured position (from the GPS data). Ac correction is applied to the predicted trajectory. The amount of correction is based on the difference between the predicted and measured trajectory as well as the predicted variance of each sensor; i.e. the higher the sensor variance, the less it is 'trusted'. In the end, the Apper was unable to reach the goal of improving the estimation; the best attempt is shown in Figure 4. The trajectory estimation from the state estimator (shown in blue) is overlaid with the raw GPS data. Every point where the two lines meet is when there is a GPS measurement (approximately once per second). Figure 4 shows that after each GPS measurement the estimation immediately diverges from the true path, only to 'snap' back when the next GPS measurement is received. The primary issue was that the sensors were very noisy. Integrating acceleration twice to obtain a position is usually a bad idea; in most applications rate sensors are used requiring a single integration. Also, the heading (obtained from the magnetometer) was very unreliable especially when the phone had non-zero roll and pitch values (see **Error! Reference source not found.**). This made for a poor predicted orientation and therefore a poor prediction of the direction of acceleration (since acceleration is measured in the phone frame, and is transformed into the inertial frame using knowledge of the orientation).

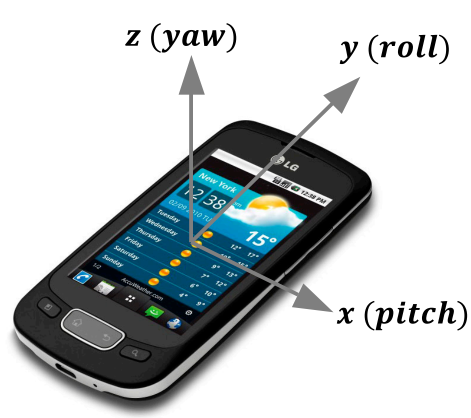


Figure 3: Device Reference Frame

If we were to begin again, more effort would have been put into determining whether developing and testing an estimation algorithm would be achievable in a three month time frame, given the team's current level of experience, which was not great.

It should be mentioned that although the Apper was unable to successfully implement an extended Kalman filter to improve position and orientation estimates it is not ruled out as a possibility. It is believed that it could be successful if the algorithm was more sophisticated. A couple suggestions for a future attempt at estimation are to use a batch-type optimization approach and to investigate how best to 'smooth' the accelerometer and orientation data. It would also be interesting to investigate how different sensors could be used to improve the algorithm. Specifically, the gyroscopes on higher end mobile devices would greatly improve the orientation estimate.

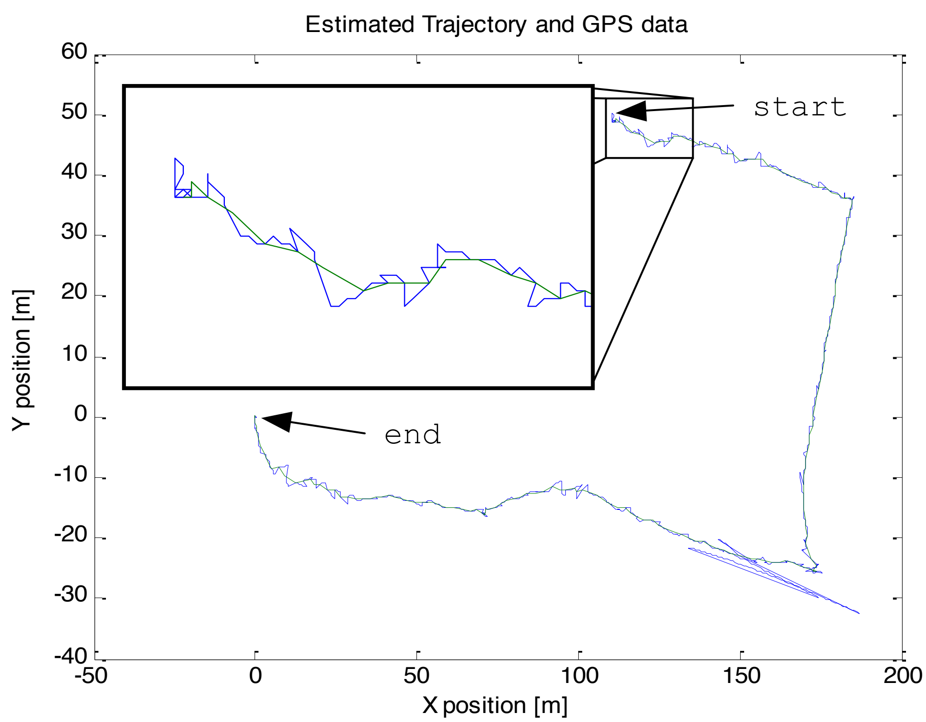


Figure 4: State estimator solution and GPS recorded trajectory overlaid

5 Member Contribution

All group members contributed to the creating the project presentations and reports. Other work was distributed as follows:

Jin Hyouk (Paul) Choi (Programmer)

Paul did the programming for the screen layouts, the data recording, and file creation. He also integrated his and Matt's parts of the code together.

Matthew Leonard (Programmer)

Matthew did all the OpenGL programming, including the 3D trajectory visualization. This included writing his own gesture library and linking them to the manipulation of the plots.

Vincent Tarantini (Apper)

Vince determined what the goal and scope of the application was going to be. He was responsible for ensuring the screens were easy-to-use, the interface intuitive, and that useful information was being presented in a meaningful way. He also attempted to implement a state estimation algorithm but was unsuccessful (see Section 4.2).

6 Apper Context

Knowledge of a vehicle's position, acceleration, or orientation as a function of time is something that is frequently required when designing and testing aerospace vehicles and the control software that operate on them. The Aerospace Sensor Suite offers aerospace students and professional, as well as radio-controlled (RC) vehicle and hobby rocket enthusiasts, a simple and accessible way of gathering motion-related data that would otherwise be very expensive and difficult to obtain.

For example, a graduate student has developed an attitude control algorithm for a monoplane and wishes to test the algorithm in real conditions under specific scenarios (e.g. inverted, large gusts, crosswind, etc.). To gather a realistic dataset, the student builds (or buys) a RC airplane which is representative of the type of aircraft she is targeting for application. She then attaches her mobile device, which is running the Aerospace Sensor Suite, to the plane. The RC plane is flown around, and is forced to experience the specific conditions of interest. Afterwards, the controller is tested by being fed sensor data from the recorded dataset and having its response examined. In this case, the Aerospace Sensor Suite offered a cheap and easy was of gathering a real dataset to use for testing an attitude controller.

The Aerospace Sensor Suite could also be used to verify the performance of a vehicle design. By knowing where an airplane was (via GPS), what it was doing (accelerometer and orientation) and when (timestamps), key aerodynamic parameters can be extracted. The vehicles maximum speed can be estimated by differentiating the GPS measurements. With knowledge of the vehicles thrust (obtained from a relatively simple static thrust test), the lift and drag coefficients can also be approximated. These data can be gathered from a full-size model of the researcher's design, or from a scaled model.

Lastly, the Aerospace Sensor Suite is a great tool for RC vehicle hobbyists who would just like to know how high and how far their vehicle went, how many g's it experienced, and how long it was in the air for. Although there are devices targeted to RC vehicle users which measure some of these things, none of them have the capability of instantly viewing the recorded data directly on the device itself, especially in such a visually pleasing and user-friendly interface.

The only things limiting this application's potential are the types of sensors on the mobile device and their accuracy. As mobile devices mature, more and more functionality will be added; this will likely include a greater variety of sensors with better performance. The iPhone 4, for example, has a 3-axis gyroscope in addition to the default accelerometer, magnetometer, and GPS sensors. Although, it is hard to predict what new sensor will become standard in all mobile devices going forward, it can be said with certainty that creative researchers will think of ways to use them for their testing or verifying their latest academic or professional aerospace design.

7 Going Forward

7.1 Future Improvements

Several features have been identified that would improve the application.

7.1.1 Data Capability

Expanding the set of recorded data to include video, audio, and gyroscopes would be a great addition. The video file may get bulky so adding in the ability to take stills either at pre-determined times or perhaps triggered by an incoming call or text would be interesting.

7.1.2 Visualization/Post-Processing

The visualization and post-processing could be improved by adding: axis labels and a time-sliding feature (allowing the user to slide an icon along the plot to get information at a specific time) to the plots, distance travelled calculation to the Points of Interest file, and the ability to overlay the ground trajectory over Google maps. Implementing a working state estimator could be a useful addition for those who are interested more accurate data. Outputting predicted aerodynamic drag and lift coefficients would also be a neat addition (although this would likely require some additional input from the user about the engine thrust, see Section 6).

7.1.3 Other Features

Some other additional features include a real-time downlink to another phone via the cellular network (or Wi-Fi for vehicles not expected to go out of a close range), automatic on/off recording triggered by acceleration measurements, and the ability to quickly publish videos and plots to social media networks.

7.2 Commercialization

The development team believes that this application could be commercialized, especially if some of the additional features outlined in Section 7.1 are implemented. However, the revenue source would likely be limited to sales of the application itself or advertising (only if the application is free).