Wirelessly Integrating Topographic Teleoperation Instrument (WITTI)

Brian Smith (bdsmith1@email.arizona.edu)
Brianna Heersink (bmheersink@gmail.com)
Alex Warren (amwarren@email.arizona.edu)

2/23/2014

Team Witty

ECE 573 - Software Engineering

Professor Sprinkle

Executive Summary

The Wirelessly Integrating Topographic Teleoperation Instrument presented in this report is an application used to visualize LiDAR data on a mobile device. This visualization will allow the user to see the environment that an autonomous vehicle is sensing and using to make decisions. The application has the functionality to visualize prerecorded data that is saved on the mobile device, prerecorded data saved on a computer, or data collected live from the Velodyne LiDAR by the computer. The application also provides the functionality for the user to select a vehicle trajectory by touching the visualized LiDAR data on the screen.

Contents

1 Project Overview **4**

I Analysis and Models **5**

2 Requirements **5**
- 2.1 All Requirements . . . 5
- 2.2 Requirements by A/B Group . . . 7
  - 2.2.1 B Requirements . . . 7
  - 2.2.2 A Requirements . . . 7

3 Application Analysis **8**
- 3.1 Overview . . . 9
- 3.2 Use Cases . . . 10
  - 3.2.1 Determine reachability of a drawn path. . . . 10
  - 3.2.2 Watching live LiDAR data . . . 10
  - 3.2.3 Demo mode viewing . . . 10

4 Domain Analysis **11**
- 4.1 Overview . . . 11
- 4.2 Refresh . . . 11
- 4.3 Settings . . . 12
- 4.4 System Algorithms . . . 13
  - 4.4.1 Network exchange . . . 13
  - 4.4.2 Vehicle Control . . . 13
  - 4.4.3 Ground Plane . . . 13
- 4.5 Phone Application Algorithms . . . 13
  - 4.5.1 3d Display . . . 13
  - 4.5.2 Path Drawing . . . 13
  - 4.5.3 Trajectory Clearance . . . 13
  - 4.5.4 Trajectory Drivability . . . 14
- 4.6 Host Computer . . . 14
  - 4.6.1 Point Reduction . . . 14
  - 4.6.2 Mesh Creation . . . 14
  - 4.6.3 Ground Mesh . . . 14
  - 4.6.4 Persistent Mapping . . . 14

II Design and Test **15**

5 Class Design **15**
- 5.1 Class Diagrams . . . 15

6 Testing Strategy **16**

7 Integration with Platform **17**

III Implementation Plan **18**

8 Task Allocation and Breakdown **18**
8.1 Breakdown . . . 18
8.2 Responsiblities . . . 18
8.2.1 Brian Smith . . . 18
8.2.2 Brianna Heersink . . . 18
8.2.3 Alex Warren . . . 18
8.3 Global/Shared Tasks and Experience . . . 19

List of Figures

1 Mock Screen Shots . . . 8
2 Phone Application State Diagram . . . 9
3 Refresh Data Sequence . . . 11
4 Change Settings Sequence . . . 12
5 App Classes . . . 15
6 Host Classes . . . 16
7 Project Timeline . . . 19

List of Tables

1 Project Overview

The Light Detection and Ranging (LiDAR) visualization application is designed to display information collected from the LiDAR on a mobile device. The app will also allow users to draw a path on top of the visualized point cloud and determine if the path is reachable.

LiDAR devices use an array of lasers to collect data that give the distance and height values of an array of points stemming out of the LiDAR that can be used to determine the physical characteristics of the surrounding landscape. Typically, LiDAR is used to help autonomous vehicles get information on their surroundings.

This app will give researchers the ability to see a subset of what the vehicle sees while the vehicle is driving or play back previously recorded data. This could be useful in debugging unexpected vehicle behavior or determining the limitations of the LiDAR when driving. This app will also help researchers visualize how and why the vehicle makes certain decisions when determining if a chosen path is reachable.

This report provides a description of the implementation details and analysis for our application. The analysis section includes a description of how the app will function, class diagrams, a description of the supported use cases, several example sequence diagrams, and a discussion of the important algorithms the application will utilize. Also included in the analysis are sample screen shots of the various user interfaces that will be part of the application. After the application analysis section there is a description of how the application will be completed, including a timeline and a separation of duties. The report concludes that the application will be completed on time and will accomplish the application goals.

Part I

Analysis and Models

2 Requirements

2.1 All Requirements

1 System

1.1 Network Exchange: The phone and host computer shall exchange information over local network.

Requirement: B
Prerequisite(s): None
Difficulty: 3

1.2 LiDAR Transfer: The phone application software shall communicate with the host computer through network to receive LiDAR data from the host computer.

Requirement: B
Prerequisite(s): 1.1
Difficulty: 1

1.3 Output Settings: The phone application software shall be capable of changing the host computer's Velodyne LiDAR data output settings through network communication.

Requirement: B
Prerequisite(s): 1.1
Difficulty: 2

1.4 Trajectory Transfer: The phone application software shall send vehicle trajectories to the host computer through network communication.

Requirement: B
Prerequisite(s): 1.1
Difficulty: 1

1.5 Manual Data Refresh: The phone application software shall be capable of refreshing the displayed Velodyne LiDAR data manually based on user input.

Requirement: B
Prerequisite(s): 2.1
Difficulty: 2

1.6 Auto Data Refresh: The phone application software shall be capable of refreshing the displayed Velodyne LiDAR data automatically through a set refresh rate.

Requirement: A
Prerequisite(s): 2.1
Difficulty: 1

1.7 Mesh Display:The phone application software shall be capable of displaying LiDAR data as a mesh.

Requirement: A
Prerequisite(s): None
Difficulty: 4

1.8 Dynamic LiDAR Driver Settings: The phone application software shall be capable of changing the LiDAR driver settings through network communication.

Requirement: A
Prerequisite(s): None
Difficulty: 4

1.9 Vehicle Control: The phone application software shall send phone trajectories to the host computer where it is converted to vehicle control commands sent to CAT Vehicle.

Requirement Level: A
Prerequisite(s): None
Difficulty: 3

2 Phone Application

2.1 Display Data: The phone application software shall load and display Velodyne LiDAR data on the phone.

Requirement: B
Prerequisite(s): None
Difficulty: 4

2.2 Draw Trajectory: The phone application software shall read a trajectory selected by the user through touch events on the phone.

Requirement: B
Prerequisite(s): 2.1
Difficulty: 4

2.3 Trajectory Clearance: The phone application software shall reject trajectories that would collide with obstacles.

Requirement: B
Prerequisite(s): 2.2
Difficulty: 3

2.4 Perspective Change: The phone application software shall be capable of changing the perspective of the displayed image on the phone.

Requirement: A
Prerequisite(s): 2.1
Difficulty: 1

2.5 Trajectory Drivability: The phone application software shall reject trajectories that are not possible to follow.

Requirement: A
Prerequisite(s): 2.2
Difficulty: 3

3 Host Computer Application

3.1 Convert Data: The host computer software shall convert the Velodyne LiDAR PCAP file to an XYZ-coordinate file with a limited complexity that will be established.

Requirement: B
Prerequisite(s): None
Difficulty: 3

3.2 LiDAR Driver Settings: The host computer software shall be capable of reducing the field of view and the range of Velodyne LiDAR data provided to the phone.

Requirement: A

Prerequisite(s): 3.1
Difficulty: 2

3.3 JAUS Compatible: The host computer software shall be capable of sending JAUS messages containing converted LiDAR data.

Requirement: A
Prerequisite(s): 3.1
Difficulty: 4

3.4 Dynamic LiDAR Conversion: The host computer software shall be capable of processing and converting the PCAP files to phone readable data as they are received within a bounded delay that will be established.

Requirement: A
Prerequisite(s): 3.1
Difficulty: 4

3.5 Ground Mesh: The host computer will generate a mesh representation of the ground, so that hills and dips in the road will be represented.

Requirement: A
Prerequisite(s): 3.1
Difficulty: 4

3.6 Persistent Mapping: The host computer will combine LiDAR data and IMU data over time to create a map that incorporates data over the course of time.

Requirement: A
Prerequisite(s): 3.1
Difficulty: 6

2.2 Requirements by A/B Group

2.2.1 B Requirements

Network Exchange (1.1)
LiDAR Transfer (1.2)
Output Settings (1.3)
Trajectory Transfer (1.4)
Manual Data Refresh (1.5)
Display Data (2.1)
Draw Trajectory (2.2)
Trajectory Clearance (2.3)
Convert Data (3.1)

2.2.2 A Requirements

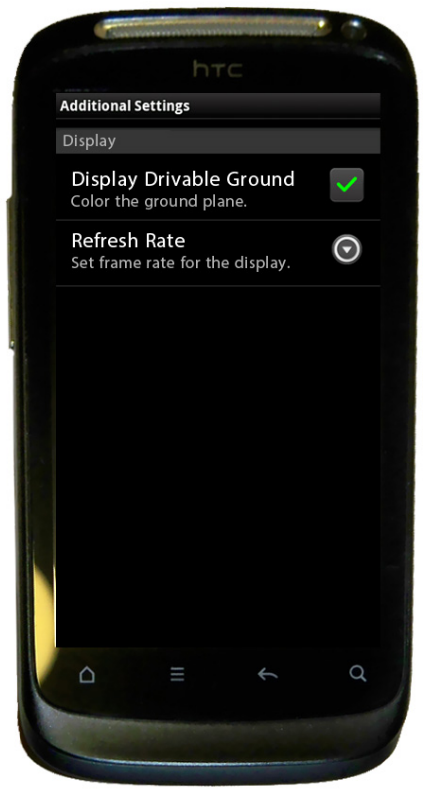
Auto Data Refresh (1.6)
Mesh Display (1.7)
Dynamic LiDAR Driver Settings (1.8)
Vehicle Control (1.9)
Perspective Change (2.4)
Trajectory Drivability (2.5)
LiDAR Driver Settings (3.2)
JAUS Compatible (3.3)
Dynamic LiDAR Conversion (3.4)
Ground Mesh (3.5)
Persistent Mapping (3.6)

3 Application Analysis

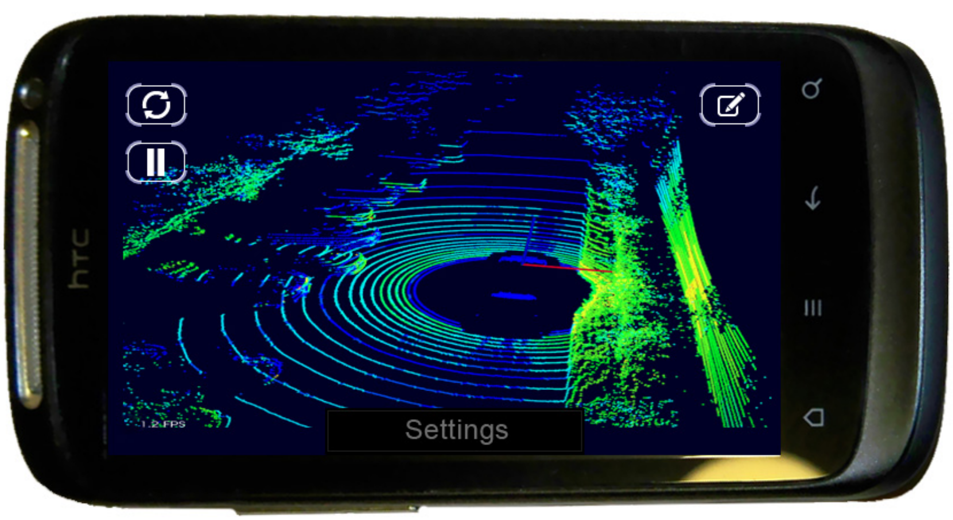
Figure 1: Mock Screen Shots

(a) Home Screen

(b) Settings Screen



(c) Display Screen



3.1 Overview

When the app initializes, the user is brought to the title screen. Here, the user is presented with three buttons which will choose the next action: Demo, Launch, and Settings. The user can navigate to the settings menu by clicking on the Settings button. The user can navigate to the display interface by clicking on the Demo button or the Launch button.

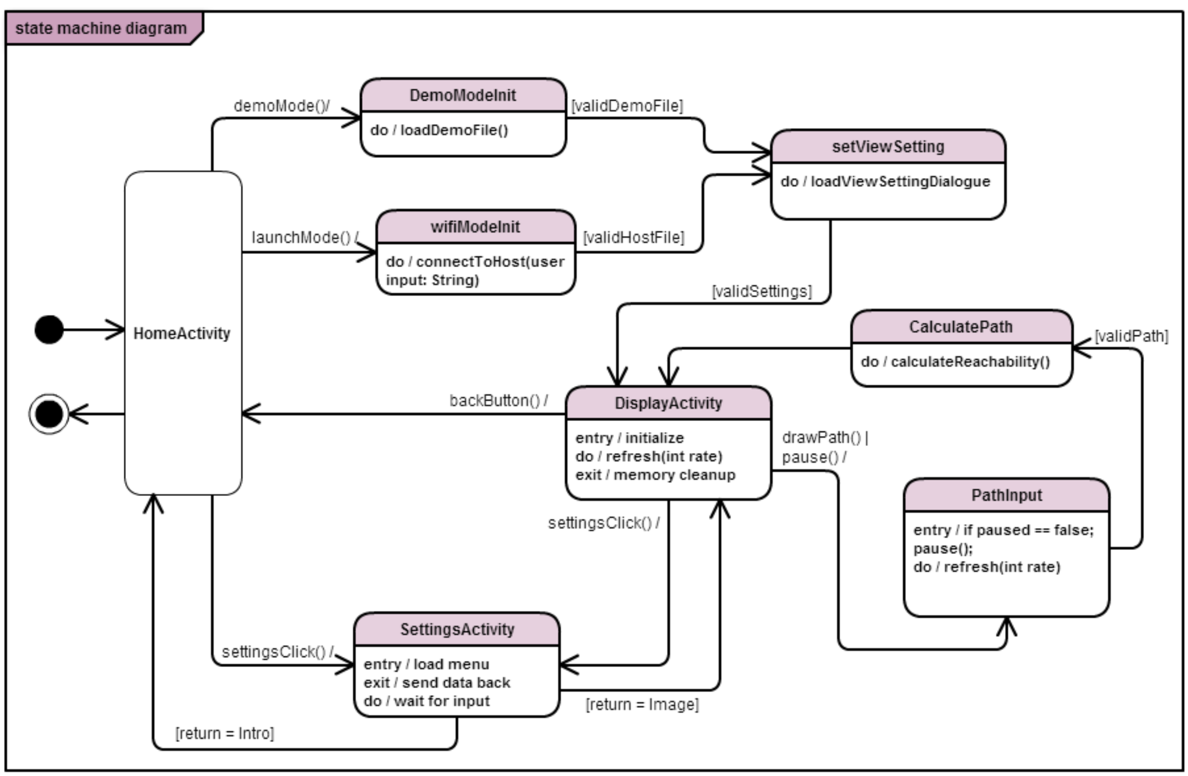
The Demo button is used initialize Demo mode, which will read point cloud data directly from a file stored on the device. The Launch button is used to initialize wireless mode, where the phone will connect with a host computer in order to receive point cloud data.

After the Demo button or Launch button is pressed, the user is presented with several settings that must be configured before moving on to the display interface. One option is the data refresh rate. This will be limited in choices, including setting the rate to 0 to refresh the display manually by tapping the screen. The other option will be to set the IP address of the host computer for wireless communication. This option will be grayed when the Demo button is used to initialize the display interface.

After setting the appropriate settings, the user is taken to the display interface where point cloud data immediately starts playing at the set refresh rate and from the set location. While the user is viewing the display interface, there will be several on-screen options. If the user is in tap to refresh mode, a draw path option and a refresh option will be available. If the user is streaming video, the only option will be to pause the playback. After a successful pause, the user will have the option to draw a path. After a valid path is inputted, the app immediately checks for reachability and informs the user of the outcome. The user is then taken back to the display interface in the paused state.

The user is able to navigate back to the title screen by clicking the back button. The user can quit the application by clicking the back button at the title screen. The home button will pause the app and return control to the device. These actions can be reviewed in the State Diagram below.

Figure 2: Phone Application State Diagram



3.2 Use Cases

The phone application has several use cases, as can be seen in the figure below. The actor for all use cases is the user of the phone application who wishes to view the LiDAR data and/or draw vehicle trajectories on a smartphone.

3.2.1 Determine reachability of a drawn path.

Summary: The app determines if the path drawn by the user on the screen is reachable, i.e. no obstacles in the path.

Actors: User

Preconditions: The app is displaying point cloud data.

Description: The user first clicks the pause button if the current mode is set to stream data. Then the user draws a path on the screen of the app. The app first checks if the path is valid, i.e. does not go out of bounds of the image. If the path is valid, the app will calculate reachability of the selected path. The app will then alert the user whether the selected path is reachable. At this point, the app will exit the path drawing state and return to the image viewing with the image paused.

Exceptions: The following exceptions are expected:

Invalid Path The user draws a path outside the boundary of the image. The app will give the use a warning popup and clear the screen.

Postconditions: The app returns to the image view screen with the current view paused.

3.2.2 Watching live LiDAR data

Summary: The user is streaming live LiDAR data.

Actors: User

Preconditions: The user is located anywhere a good connection between the device and the host computer can be established. The app is open to the title screen, HomeActivity.

Description: The user selects Launch Mode. The user selects either a constant refresh rate or tapping the screen to get the current LiDAR view. The user enters the host computer's IP address and selects to get live LiDAR data. Once the user acknowledges all settings, a wireless connection between the device and host computer will be established. The device will then begin streaming live LiDAR data.

Exceptions: The following exceptions are expected:

Connection Signal Lost The connection between the host computer and the device is lost. The app will do some memory clean up and then return to the Introduction screen.

Postconditions: The app is open to the DisplayActivity.

3.2.3 Demo mode viewing

Summary: The user is viewing prerecorded LiDAR data stored on the mobile device.

Actors: User

Preconditions: The app is open to the title screen.

Description: The user selects Demo Mode. The user selects either a constant refresh rate or tapping the screen to update the LiDAR view. Once the user acknowledges all settings, the app beigns displaying the prerecorded LiDAR data.

Exceptions: None.

Postconditions: The app is open to the DisplayActivity.

4 Domain Analysis

4.1 Overview

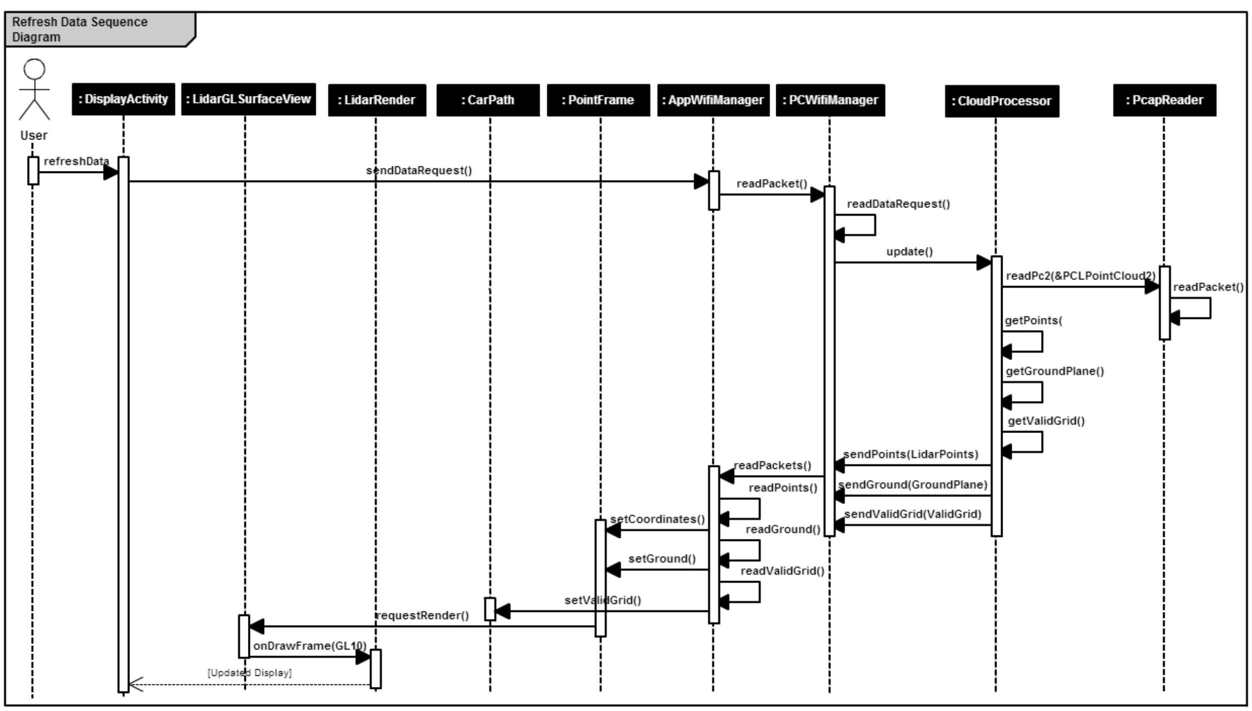
The application will involve many processes transparent to the user. In addition to the software on the phone, hardware and software outside of the phone are involved in providing the full functionality on the Android app.

4.2 Refresh

The main functionality of the app is to provide the user a visualization of the data collected and saved from the Velodyne LiDAR. The objects and methods involved in refreshing the image displayed on the phone can be seen in the sequence model below. The preconditions to this sequence model are as follows:

1. The app is in the Launch mode rather than the Demo mode.
2. The app settings are configured to have a refresh rate of 0, which requires the user to manually select when the screen should be updated.
3. The app settings are configured to have the LiDAR data read from a PCAP file on the computer's hard drive.

Figure 3: Refresh Data Sequence



This sequence model shows the objects involved in getting new LiDAR data from the computer and rendering the new image on the app display. This involved wireless communication for the data request as well as the LiDAR data between the phone and the computer.

On the computer side, the data is read by the PcapReader. The CloudProcessor then converts the data into a point cloud and reduces the point cloud sufficiently for the phone to be able to render the data relatively quickly. The point cloud reduction methods include reducing the field of view of the data to only include points in front of the vehicle and using the Point Cloud Library's VoxelGrid filter to downsample the dataset. Additionally, the CloudProcessor will determine a GroundPlane based on the LiDAR data. This GroundPlane can be used on the phone app in the Draw Path use case in order to determine the car's path trajectory on the ground based on the user's touch input. Finally, the CloudProcessor will establish a ValidGrid based on the dataset. This ValidGrid will identify areas that the vehicle will be able to drive over since there are no large objects above the GroundPlan in these areas.

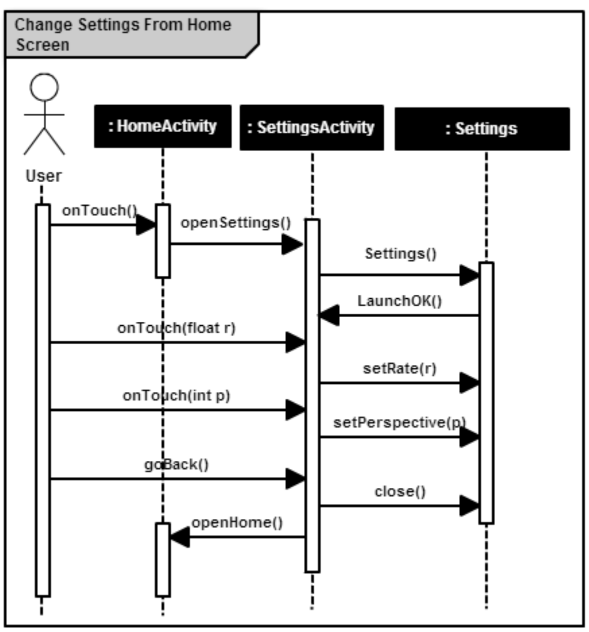
After the data is read and processed, the PcWifiManager will send the three datasets, LidarPoints, GroundPlane, and ValidGrid, to the AppWifiManager using the connection that was established upon initializing the Wi-Fi communication between the computer and the phone.

The phone will then save the LidarPoints, GroundPlane, and ValidGrid. The LidarSurfaceView object will call the LidarRender object to render the image using the updated data received from the computer.

Another functionality of the app not shown in the sequence diagram above is sensing the user's touch trajectory for the vehicle. This occurs after the user selects to draw a new path trajectory for the vehicle. This will involve calculating the coordinates on the GroundPlane and determining whether these coordinates all fall within the ValidGrid. If the path selected by the user leaves the ValidGrid, then the app will display an error message using the Alert object. If the path falls within the ValidGrid, then the translated path coordinates will be sent to the computer through the AppWifiManager and the app will display a notification indicating success using the Alert object.

4.3 Settings

Figure 4: Change Settings Sequence



The user is currently viewing the HomeActivity user interface. The user then taps the Settings button which will use the Android touch callback method to call the openSettings() method. This will initialize and create the SettingsActivity menu. In the SettingsActivity menu, the user can make changes to the different settings, including changing the refresh rate and changing the perspective. The SettingsActivity will also use the Android touch callback methods to determine where the user touches and how to respond appropriately, which includes sending messages to the Settings class that holds the values for the various settings during the entire application lifetime. After the user has finished with the SettingsActivity, the back button will be used to initialize the HomeActivity and return the user interface to the title screen.

4.4 System Algorithms

4.4.1 Network exchange

The phone and host computer will communicate using Wi-Fi and a server host relationship. Because the host computer will be running software programmed in C++ and the phone will be running Andriod, this may require developing code to manually marshal and demarshal data. We will develop a standard set of messages with expected header information for socket communication.

4.4.2 Vehicle Control

With proper calibration of the LiDAR to real world distances, it is possible to map a drawn path to a series of GPS coordinates or to a path in relative coordinates. The path following code that is currently developed for the CATVehicle does not allow messages to be transferred directly, but it is possible to create a JAUS message that conveys that information or to have spawn the process with a datafile created to be loaded as it starts.

4.4.3 Ground Plane

The ground plane will be determined by first selecting the points nearest the vehicle that are not on the vehicle itself. The RANSAC algorithm will be used to test possible planes by calculating the number of matching points and determining an estimate for the local terrain. It can be assumed that the four tiresof the vehicle are on a level plane, so any far deviation from this standard will be considered an error.

4.5 Phone Application Algorithms

4.5.1 3d Display

The LiDAR visualization will be accomplished using the OpenGL es2 libraries for Android. There are several different options depending on what we finally display. The ground plane can be displayed as a simple mesh with coloring based on the occupancy grid, which can be done using vertex coloring and smooth interpolation. The points can be displayed as sprites, and if we choose to create a mesh, this will need to be converted from the representation in point cloud library to one that can be loaded in OpenGL.

4.5.2 Path Drawing

The path will be drawn onto the ground plane from a 3d perspective. Because the camera is represented as a matrix that transforms from 3d position to a 2d pixel coordination, it is possible to use linear algebra to find a ray cast from the center of each pixel into the 3d space. After finding this line, it is possible to find its intersection with the ground plane. As the user drags a path accross the map, it may be desirable to smooth the path to prevent noise from making the path undrivable. The path can be displayed as a series of points or as a mesh.

4.5.3 Trajectory Clearance

Clearance of a trajectory will be determined by checking if there is space of a predetermined amount on both sides of the path. To facilitate this, the host computer will send an occupancy grid that it calculates based on a check of whether points exist above each sector with the ground plane as a reference. This will be a grid with a predetermined size in meters and indexing scheme. To save space in transfer, the open sectors will be sent as an array to the vehicle. A simple check would be to see if each point in that path is clear in a radius of sectors.

4.5.4 Trajectory Drivability

To determine if a path is reachable from a control point of view, we will assume that the vehicle is traveling below a safety speed. From this it is possible to assume that a minimum turn radius will be reachable. As long as the path, when suitable interpolated and smoothed, does not have tighter turns that this, the path is determined to be reachable. This problem could easily be expanded to include issues of under and over shoot.

4.6 Host Computer

4.6.1 Point Reduction

Point reduction can be accomplished in many ways. The first method is to limit the output of the Velodyne by setting speed, field of view, and range. Next it is possible to programmatically ignore points that are deemed unnecessary. One option for reducing the number of points is to use a voxel grid to reduce the resoultion of the 3d space. This can be done using the Point Cloud Library (PCL). Another possibility is to randomly discard data points.

4.6.2 Mesh Creation

The PCL library has a functionality that creates a mesh by gready triangluation where close points are assumed to lie on the same surface. This could be used to create a mesh, although because objects are occluded in a static frame, the mesh would be missing backs. We could look into patching the holes or extended the mesh away from the center. Also, with persistent mapping it would be possible to refine the mesh with more information as the car moves.

4.6.3 Ground Mesh

Similarly to the ground plane, we can use RANSAC to test possible planes and then check for accuracy. It is also possible that mesh creation will create a ground mesh as a side effect, although we would need to extract it from the other objects. This could be accomplished by starting with the car's natural ground plane and extending outwards by adding nearby triangles.

4.6.4 Persistent Mapping

Persistent Mapping would be challenging due to timing issues where a small difference in angle can change a distant point's position by a large margin. This is an application of Simultaneous Localization and Mapping. It would be necessary to receive data from the Inertial Measurement Unit (IMU) regarding orientation and then use this probabilistically to update the point cloud. Further reseach would be necessary to determine the best approach.

Part II
Design and Test

5 Class Design

5.1 Class Diagrams

Figure 5: App Classes

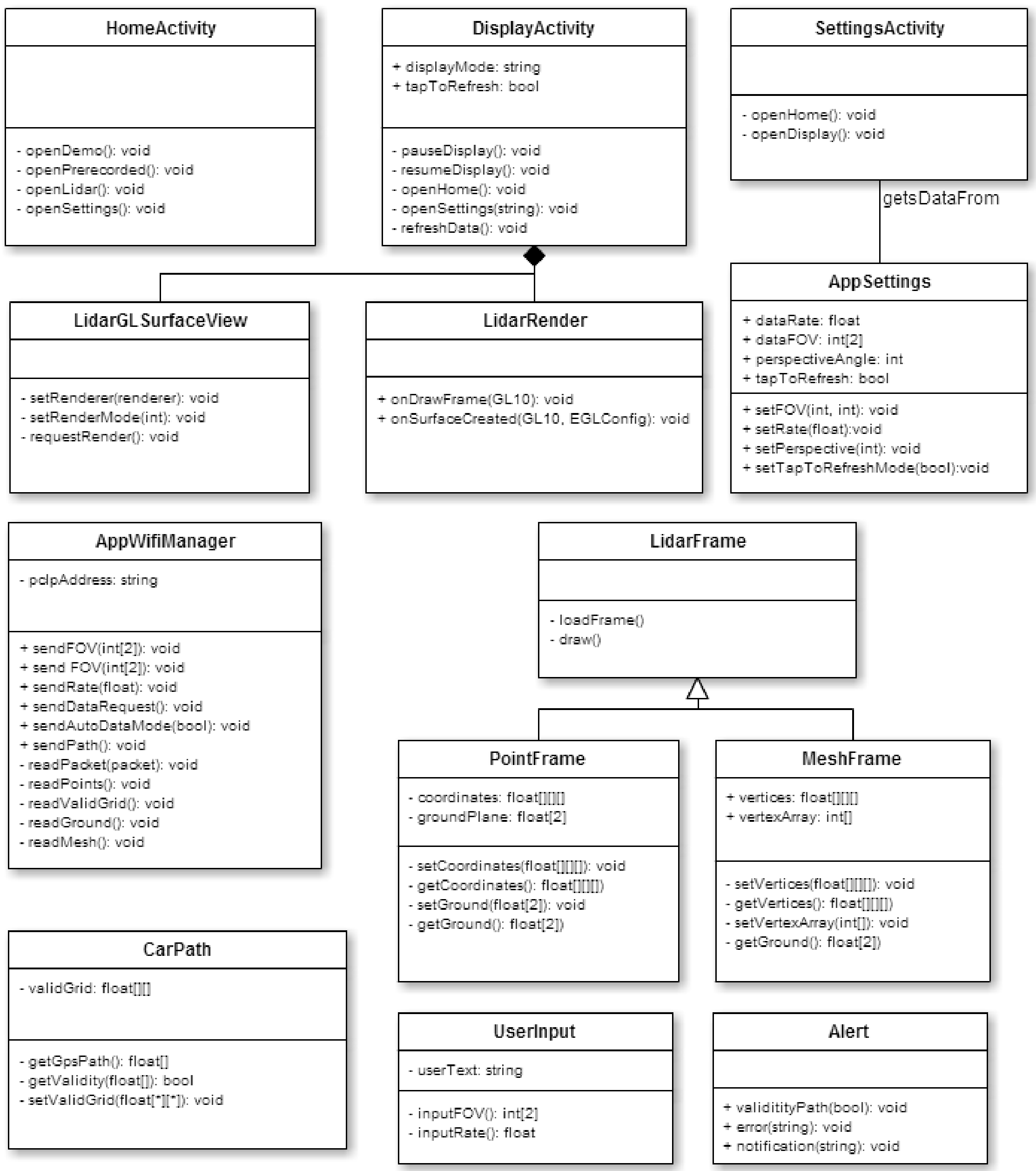
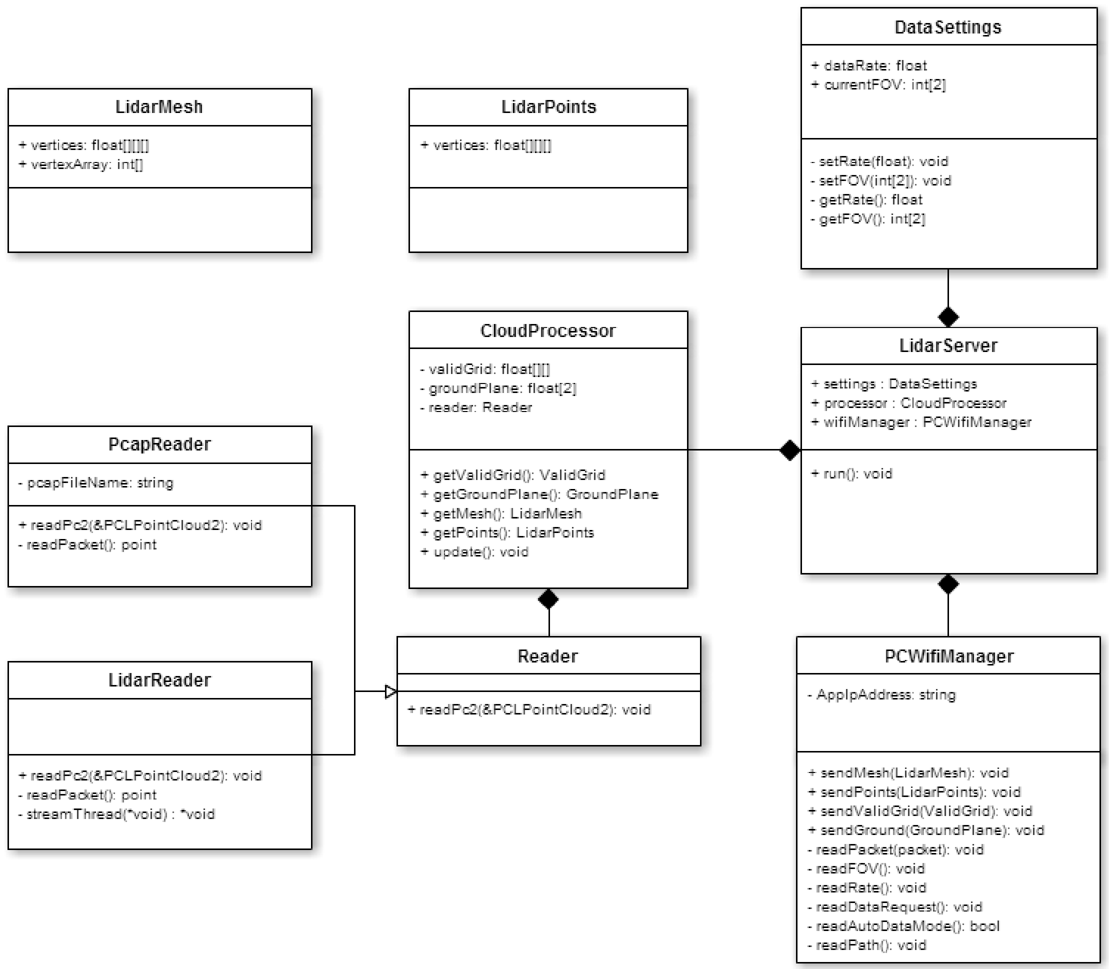


Figure 6: Host Classes



6 Testing Strategy

1 System

1.1 The phone sends a message across network and the host computer responds. If any of the remaining system tests pass, this passes.

1.2 The phone requests LiDAR data test over network and response is sent back over network. The phone checks received data matches stored test LiDAR.

1.3 A set settings message is sent from the phone to the host computer and successfully changes the settings to a comprehensive set of configurations.

1.4 The phone sends vehicle trajectory test to host computer, and the host checks received data matches stored test trajectory.

1.5 The phone sends refresh request message and receives complete LiDAR data sets.

1.6 The phone sends a refresh rate request message, and it receives complete LiDAR data sets at the requested interval.

1.7 LiDAR is converted to a data format that can be displayed as mesh. Change driver settings commands are sent from the phone to the host computer, and the driver is restarted with the new settings.

1.8 A test trajectory is sent from the phone to the host where it results in the simulator moving through the trajectory.

2 Phone Application

2.1 An already prepared mesh or point cloud intermediary message is stored on the phone and can be loaded into OpenGL and drawn.

2.2 The phone will generate reasonable paths and determine the user input required to create them. It will simulate this user input and determine if the created path is similar to the initial one.

2.3 In combination with part 2.2, paths can be generated that are both good and bad. The phone will respond to the simulated inputs with behavior appropriate to the path. The original paths will be created independently by hand using another method to cross verify the results This is all be done with test LiDAR data in 1.2.

2.4 The phone will simulate user input to change the perspective, and the OpenGL representation of a camera will change to expected values or in the expected way. This will depend on the implementation of the perspective changing.

2.5 The test for requirement 2.3 will be extended to check if paths that meet drivability requirements based on independently created and judged paths

3 Host Computer Application

3.1 Using a reference file of PCAP data converted using a Matlab script, the host software will compare its output to the reference, and may allow some difference in the number of points converted.

3.2 The host software will check that the data being received from the LiDAR is consistent with the settings given to the LiDAR in terms of range and angle.

3.3 A JAUS component will be created that will perform the checks in test 3.1 given messages it receives from the JAUS compatible LiDAR component.

3.4 With components that pass the requirement 3.1, the host software will be able to convert the LiDAR data within a predetermined time limit.

3.5 A test dataset will be created of fake points from a surface and obstacles and noise. The test data will be fed into the host computer and the resulting ground mesh compared to the surface that generated it in order to check if error falls within a threshold.

3.6 Given a prerecorded set of data for the LiDAR and IMU, the mapping process will create a map that is consistent across time and successfully maps a known object as the car drives around it.

7 Integration with Platform

The application will be interacting with the platform's Wi-Fi in order to communicate with a host computer. When the application starts the wireless mode, an initialization process between the host computer and the device is launched, which includes instantiating an instance of AppWifiManager. The prerequisite for this is that a PCWifiManager will already be running on the host computer waiting for a connection message from the application.

After a connection is made, the application will then automatically download new data from the host computer based on the set refresh rate. If the refresh rate is set to zero, the app will only request new data from the host computer when the user taps the screen, indicating a refresh action. The Wi-Fi manager inside of the app will communicate directly with the view activity that is currently in focus, since that is how user input will be managed, i.e. each view activity will have their own versions of the touch callback methods. There are several methods that will be implemented that will help send messages between the AppWifiManager and the current activity, which can be directly seen on the class diagram.

Part III
Implementation Plan

8 Task Allocation and Breakdown

8.1 Breakdown

Broadly, the tasks are broken down into three categories as follows: computer software, wireless interface, and application software. Although task allocation may vary throughout the project, the initial allocations for the "B requirements" are listed below. The "A requirements" allocation will be determined upon finalization of which requirements will be implemented.

8.2 Responsiblities

8.2.1 Brian Smith

Domain: Computer Software

Requirements: 1.5, 2.3, 3.1

Classes: LidarServer, CloudProcessor, LidarPoints, Reader, PcapReader

8.2.2 Brianna Heersink

Domain: Wireless Interface and Data Settings

Requirements: 1.1, 1.2, 1.3, 1.4, 1.5

Classes: PcDataSettings, PcWifiManager, AppWifiManager, AppSettings, SettingsActivity

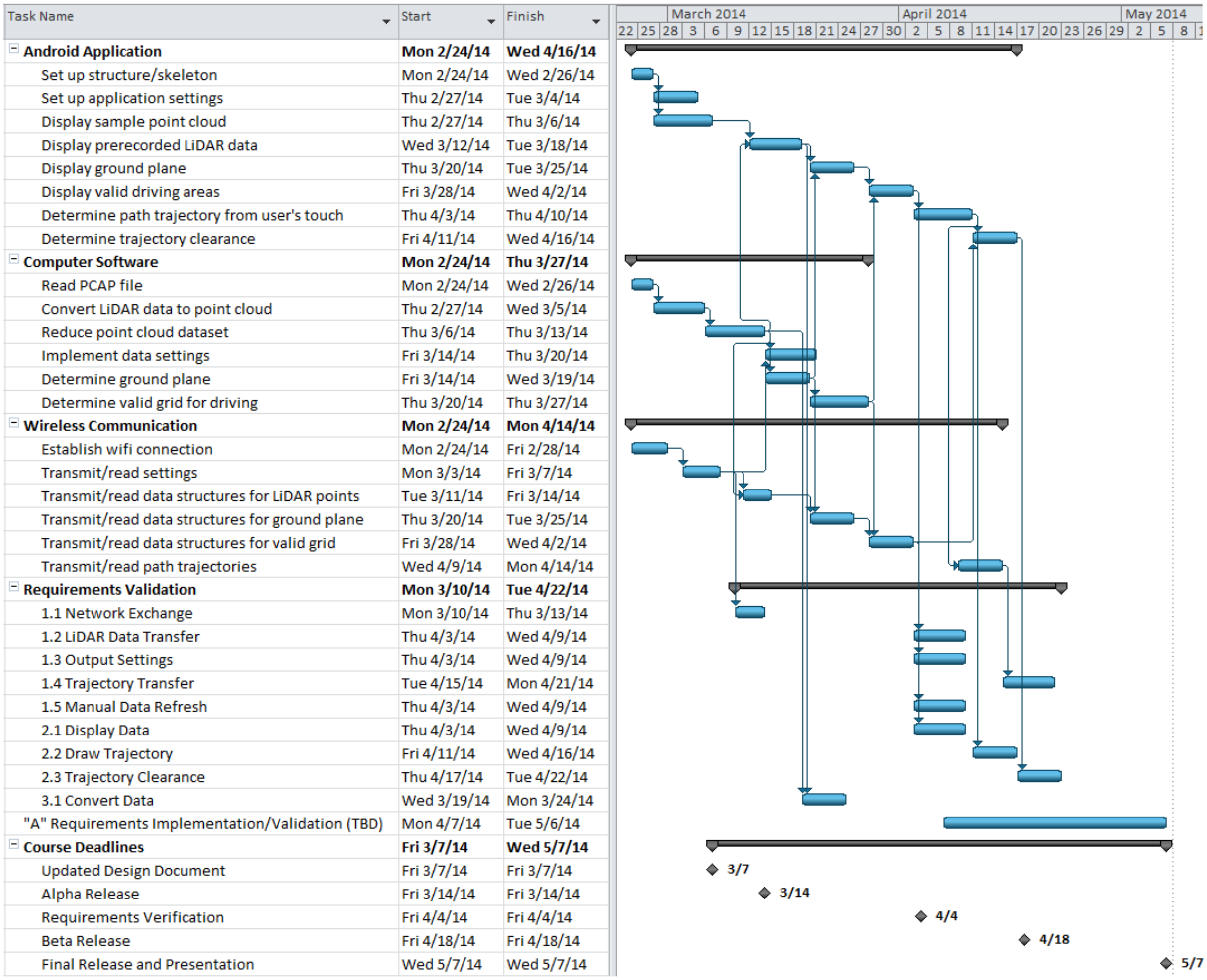
8.2.3 Alex Warren

Domain: Application Software

Requirements: 1.5, 2.1, 2.2, 2.3

Classes: HomeActivity, DisplayActivity, LidarGLSurfaceView, LidarRender, LidarFrame, PointFrame, CarPath, UserInput, Alert

Figure 7: Project Timeline



Task Name	Start	Finish
Android Application	**Mon 2/24/14**	**Wed 4/16/14**
Set up structure/skeleton	Mon 2/24/14	Wed 2/26/14
Set up application settings	Thu 2/27/14	Tue 3/4/14
Display sample point cloud	Thu 2/27/14	Thu 3/6/14
Display prerecorded LiDAR data	Wed 3/12/14	Tue 3/18/14
Display ground plane	Thu 3/20/14	Tue 3/25/14
Display valid driving areas	Fri 3/28/14	Wed 4/2/14
Determine path trajectory from user's touch	Thu 4/3/14	Thu 4/10/14
Determine trajectory clearance	Fri 4/11/14	Wed 4/16/14
Computer Software	**Mon 2/24/14**	**Thu 3/27/14**
Read PCAP file	Mon 2/24/14	Wed 2/26/14
Convert LiDAR data to point cloud	Thu 2/27/14	Wed 3/5/14
Reduce point cloud dataset	Thu 3/6/14	Thu 3/13/14
Implement data settings	Fri 3/14/14	Thu 3/20/14
Determine ground plane	Fri 3/14/14	Wed 3/19/14
Determine valid grid for driving	Thu 3/20/14	Thu 3/27/14
Wireless Communication	**Mon 2/24/14**	**Mon 4/14/14**
Establish wifi connection	Mon 2/24/14	Fri 2/28/14
Transmit/read settings	Mon 3/3/14	Fri 3/7/14
Transmit/read data structures for LiDAR points	Tue 3/11/14	Fri 3/14/14
Transmit/read data structures for ground plane	Thu 3/20/14	Tue 3/25/14
Transmit/read data structures for valid grid	Fri 3/28/14	Wed 4/2/14
Transmit/read path trajectories	Wed 4/9/14	Mon 4/14/14
Requirements Validation	**Mon 3/10/14**	**Tue 4/22/14**
1.1 Network Exchange	Mon 3/10/14	Thu 3/13/14
1.2 LiDAR Data Transfer	Thu 4/3/14	Wed 4/9/14
1.3 Output Settings	Thu 4/3/14	Wed 4/9/14
1.4 Trajectory Transfer	Tue 4/15/14	Mon 4/21/14
1.5 Manual Data Refresh	Thu 4/3/14	Wed 4/9/14
2.1 Display Data	Thu 4/3/14	Wed 4/9/14
2.2 Draw Trajectory	Fri 4/11/14	Wed 4/16/14
2.3 Trajectory Clearance	Thu 4/17/14	Tue 4/22/14
3.1 Convert Data	Wed 3/19/14	Mon 3/24/14
"A" Requirements Implementation/Validation (TBD)	Mon 4/7/14	Tue 5/6/14
Course Deadlines	**Fri 3/7/14**	**Wed 5/7/14**
Updated Design Document	Fri 3/7/14	Fri 3/7/14
Alpha Release	Fri 3/14/14	Fri 3/14/14
Requirements Verification	Fri 4/4/14	Fri 4/4/14
Beta Release	Fri 4/18/14	Fri 4/18/14
Final Release and Presentation	Wed 5/7/14	Wed 5/7/14

8.3 Global/Shared Tasks and Experience

The integration of all components will be a shared task between all group members. Additionally, all members will write tests, either for their own requirements or for other's requirements depending on the individual workloads. Finally, all members will review each other's code and contribute to required documentation that is submitted throughout the project.

The task allocation was determined to allow the work to be divided and then and integrated easily. Alex was interested in gaining more experience with OpenGL, so his task allocation included working on the data visualization. The remaining tasks were split between Brian and Brianna.