Automated Filtering of Non – Terrestrial Points From a LiDAR Point Cloud Data Using a Python Script

By
EDGARDO V. GUBATANGA JR.
2006 – 30695

Submitted to the Department of Geodetic Engineering
College of Engineering
University of the Philippines

In Partial Fulfilment of the Requirements
For the Degree of Bachelor of Science
In Geodetic Engineering

College of Engineering
University of the Philippines Diliman
Quezon City

2014

ACKNOWLEDGEMENT

First of all, I would like to thank God for making it possible for me to finish this research even though I missed some church masses to do things for my thesis. I would also like to thank my parents for their prayers, support and patience for they are the ones who told me to never give up and to have faith in God.

I would like to thank my faculty adviser, Engr. Rosario Ang, for all the advice, support and assistance given to me despite her busy schedule. I am also very thankful to Dr. Ariel Blanco for his support and suggestions for this research. I will never be able to finish this research without them.

I am thankful to UP DREAM, especially to Dr. Enrico Paringit, for providing the 12 sample LAS file data used in this research. I would like to thank Ms. Simonette Lat and Ms. Joida Prieto for assisting me in processing the data request. I also give thanks to Engr. John Louie Fabila for his concern and suggestions regarding the research. I also thank Engr. Sarah Jane Samalburo for allowing me visit my adviser even during office hours.

I give special thanks to Engr. Marcus Patam, for giving me substantial suggestions that helped me finish this research. He greatly influenced this thesis. I also thank Engr. Kenneth Langga and Mr. Jeffrey Delica for sharing their knowledge for this research.

I am thankful for my housemates: Ms. Joanne Balaga, Ms. Carlyn Ibanez, Mr. Emmanuel Ricohermoso, Ms. Ailyn Olanda, Ms. Elainne Lopez, Mr. Bobmar Prado, Mr. Rey Jalbuena and Mr. Kenneth Rodriguez, for their support during the creation of this research.

I would like to thank Mr. Martin Isenburg for sharing his knowledge about LAS files.

And last, I would like to give thanks to my Toshiba NB250 netbook, which I used for this research, for not breaking down despite its age and measly computing power.

ABSTRACT

Airborne laser scanners gain popularity in land survey due to its fast acquisition of spatial information in a form of Laser Detection and Ranging (LiDAR) point cloud data. These point cloud data, if properly processed, can provide spatial information regarding the bare earth, vegetation, buildings and other man-made structures. However, the point cloud data acquired by the airborne laser scanners are prone to contamination of non-terrestrial points. These non-terrestrial points are characterized by being suspended in the 3D space (either isolated or in small clusters) with no apparent spatial connection to the ground. These are commonly caused by clouds, birds or low-altitude aircrafts. If not removed, digital terrain models (DTMs) generated from a contaminated point cloud data will display an erroneous terrestrial surface. In the UP Disaster Risk Exposure and Mitigation (UP DREAM) program of the Department of Science and Technology (DOST) and UP Training Center for Applied Geodesy and Photogrammetry (UP TCAGP), the acquired point cloud data are intermittently contaminated with non-terrestrial points. Personnel from the DREAM program manually detects and filter these non-terrestrial points from the point cloud data which consumes significant amount of time from the workflow. A new method of filtering non-terrestrial points is proposed using the Python programming language and with the aid of the Laspy module. Comparing the number of deleted non-terrestrial points using the created Python script against those deleted manually, the automated filtering has an accuracy of >90%.

KEYWORDS: LiDAR, non-terrestrial points, filtering, Python

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Light Detection and Ranging (LiDAR) technology is applied in airborne laser scanners for an efficient gathering of spatial information. LiDAR data covers information of the ground terrain as wells as man-made structures and vegetation (Liu, 2008). Modern computer technology allows the application of computer algorithms capable of segregation of ground terrain, structures and vegetation in a LiDAR data (Baltzavias, 1999). However, LiDAR are prone to contamination of points that cannot be classified either as ground, vegetation or man-made structures. These points, by visual inspection using LiDAR-displaying software such as TerraScan or LAStools, are observed to be suspended in mid-air, either isolated or in small clusters, and has no apparent spatial connection to the ground. These points may be referred generally as non-terrestrial points since they represent no specific kind of physical object. The usual causes of these points are clouds, birds, haze or low-flying aircrafts (Meng, 2010) (Watershed Sciences, 2006). Non-terrestrial points can void the validity of the contaminated data for almost any usage. Thus, filtering of non-terrestrial points is a necessary part of a LiDAR processing workflow.

In the case of the UP Disaster Risk Exposure Assessment and Mitigation (UP DREAM), a project of Department of Science and Technology (DOST) and UP Training Center for Applied Geodesy and Photogrammetry (UP TCAGP), LiDAR data gathered by the Data Acquisition Component (DAC) of UP DREAM, like any other raw LiDAR data, are occasionally contaminated with non-terrestrial points. Therefore, one of the tasks of the Data Processing Component (DPC) is to filter these non-terrestrial points. Personnel of DPC manually scan each 1km * 1km block of LiDAR point cloud data for non-terrestrial points using the TerraScan

software. The personnel subjectively classify which points are considered non-terrestrial. Then, using the same software, points flagged as non-terrestrial are deleted by enclosing them in a manually drawn polygon and allowing the software to delete all the points inside it. Scanning and filtering of a single 1km * 1km block takes at least a minute to perform. This process is iteratively performed towards all LiDAR blocks, thus consuming a large portion of the DPC workflow. Tools from TerraScan and LAStools are occasionally used to automatically filter out non-terrestrial points from LiDAR blocks but they do not yield optimal results. Therefore, abandoning the concept of manual filtering is not an option.

This research proposes a new method of filtering non-terrestrial points automatically. The goal is to create a Python script that automatically detects and deletes non-terrestrial points based on an elevation histogram. The script aims to finish its task in a minute or less, shorter than the duration of manual filtering, or else, it will make the DPC workflow longer than its current duration. The script must also have accurate filtering against the manual filtering and must also work on areas with high slope.

CHAPTER 2

REVIEW OF RELATED LITERATURE

2.1 Light Detection and Ranging (LiDAR)

LiDAR technology has a capability to acquire large amount of spatial data across different areas. This remote sensing technology is classified as an active form of remote sensing because the system transmits its own light pulses (with 1064nm laser wavelength for both ALTM Pegasus and ALTM Gemini, which are both used by UP DREAM) instead from other sources like the sun and measures the time it takes for the pulse to return to the sensor, which exhibits the same concept with radar. LiDAR platforms can either be airborne, ground-based mobile or ground-based stationary. However, for this study, only LiDAR data from airborne platform are used. LiDAR data is known to have a very high spatial accuracy over its point cloud data. This makes the LiDAR data suitable for the generation of 3D terrain and surface models.

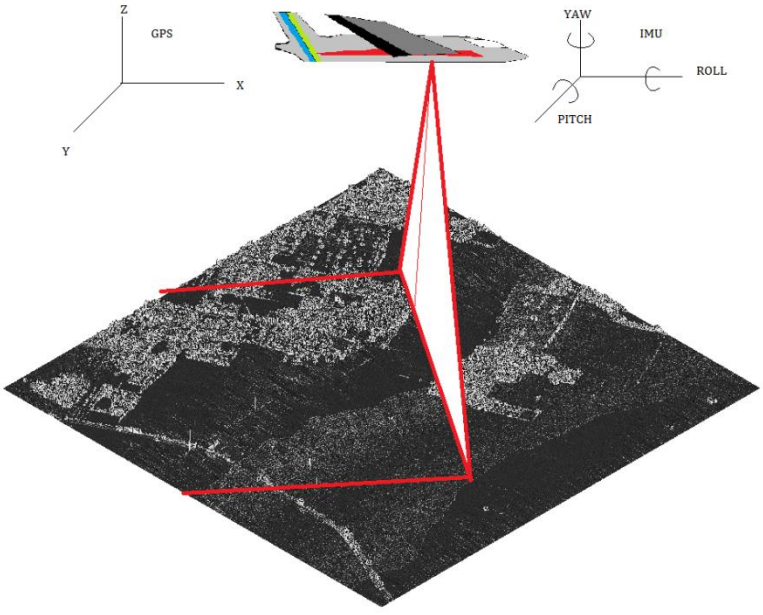

Figure 2.1 Diagram showing how airborne laser scanners operate

The concept of how LiDAR technology gathers spatial data is not much complex. The LiDAR instrument measures the duration that it takes for the light pulse from the instrument to hit an object and reflect back to the sensor. Using the recorded duration, the distance between the instrument and the hit object is calculated. In case of the airborne platform, The airplane has a Global Positioning System (GPS) installed onboard. Therefore, using the position of the airplane, the recorded laser angle and the distance between the airplane and the object, the actual position of the object is obtained, as shown in Figure 2.1.

However, despite of this simple concept, the process turns out to be more complex due to demands for high accuracy. The speed of the airplane, the turbulence during flight and the fact that the LiDAR instrument transmits at least 100,000 light pulses per second challenges the maintenance of high accuracy of the LiDAR data. With the aid Inertial Measuring Unit (IMU) installed, highly accurate positions of LiDAR points are attainable. The IMU has the capability to measure the motion of the airplane along the three (3) orthogonal directions which contributes to the determination of the airplane's exact position. Aside from the GPS onboard the airplane, there is also a GPS set up on a ground point with a known position. The GPS measurement at the ground station provides corrections for the measurements made by the GPS onboard the airplane.

Aside from high light pulse rate (up to 400,000 pulse per second for Optech ALTM Pegasus), multiple returns from each single pulse contributes to a very high point density of a LiDAR point cloud data (Schmid, et al, 2008). This information provides great contribution in point classification (Mallet, et al, 2011). Vegetations have a greater probability to yield multiple returns compared to bare-earth and man-made structures which is usually opaque where the light pulse cannot pass thru.

2.2 LAS File and Laspy

LAS stands for laser file format. It is a public file format developed for the exchange of three dimensional 3D LiDAR point cloud data among the users. Aside from the LiDAR data, LAS also support other form of 3D point clouds that contains at least X, Y and Z coordinates. Aside from the Cartesian coordinates, a LAS file may also contain additional information regarding each point. In the case of LiDAR data, example of these additional information are number of returns, intensity and GPS time.

The latest version of LAS file so far is LAS 1.4 which is approved by the American Society for Photogrammetry and Remote Sensing on November 14, 2011 (ASPRS, 2013). However, the older version which is LAS 1.2 (approved September 2, 2008) is more popularly applied and is also used for this research. LAS 1.2 has two (2) advantages against the previous versions LAS 1.0 and LAS 1.1. The first one is using GPS Absolute Time aside from the GPS Week Time. Using only GPS Week Time may cause errors because GPS time stamps are reset during Saturday midnight. This error is eliminated in LAS 1.2. Another advantage of LAS 1.2 is the incorporation of Red, Green and Blue image data for each point in the LiDAR point cloud data. However, these image data is not available in the sample data used in this research.

Each LAS file is composed of three (3) parts, namely: public header block, variable length records and point data records. The public header block contains information regarding the file in general like number of points coordinate boundary of the LAS file. The variable length records contain metadata such as projection in formation and the user application data (ASPRS, 2008).

Table 2.1 Items present in the point data record of a LAS file

Item	**Definition**
X	X coordinate saved as integer. Multiplied to scale then added to offset value to generate actual X coordinate value
Y	Y coordinate saved as integer. Multiplied to scale then added to offset value to generate actual Y coordinate value
Z	Elevation saved as integer. Multiplied to scale then added to offset value to generate actual elevation
Intensity	The magnitude of the pulse return reflection.
Return Number	Return number for the pulse since a single pulse may yield multiple returns
Number of Returns	Number of returns received from a single pulse
Scan Direction Flag	The direction of the scanner mirror during the transmission of the pulse.
Edge of Flight Line	Equals to "1" if the point is located to the scan's end.
Classification	Contains classification values representing different categories. Equal to "0" in unclassified.
Scan Angle Rank	Angle where light pulse was transmitted from the instrument
User Data	Empty item dedicated for user's discretion
Point Source ID	Indicates the file where the point came from

The point data record contains the information regarding each point in the point cloud from the LAS file. These large amount of data can be manipulated using a Python script with the help a module named Laspy which is created by Grant Brown. Laspy is a free Python library that can read, manipulate and write LAS files (Brown et al, 2012). Laspy 1.2.5, its latest version, is able to support LAS formats 1.0 up to 1.4. One of the classes belonging to the Laspy library is the File class which reads the point records. Laspy converts point data record into point a numpy array. Therefore, installation of Numpy library is necessary alongside with the installation of Laspy.

2.3 Filtering of Non-terrestrial Points

Literatures pertaining to the deletion of non-terrestrial points is very scarce because non-terrestrial points are not always present in a LiDAR point cloud data. More researches focus instead on the filtering of vegatation and structures to create a bare-earth surface which

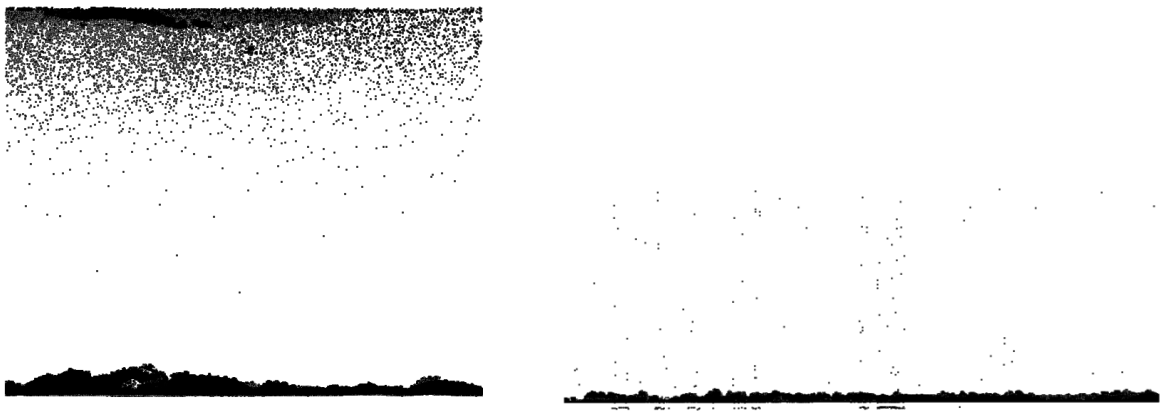

Figure 2.2 Cross-sectional views of Cag21D and Agn5M respectively. Cag21D has dense and clustered non-terrestrial points. Agn5M has dispersed non-terrestrial points.

will be used for the digital terrain model (DTM) generation (Mongus et al, 2013)(Li et al., 2013). Though non-terrestrial points are not always present, there could normally be around fifty (50) up to one hundred (1000 non-terrestrial points present for every seven and a half (7.5) up to nine (9) million LiDAR points (Watershed Sciences, 2006). The term "non-terrestrial" is from (Watershed Sciences, 2006) and is adopted for this research. These non-terrestrial points are usually caused by birds, clouds, haze and vapour.

CHAPTER 3

METHODOLOGY

3.1 Input File

The LiDAR data generated by UP DREAM is acquired by its DAC who uses three (3) different airborne laser scanners, namely, ALTM Pegasus, ALTM Gemini and ALTM Aquarius, which are all manufactured by Optech. The sample data used in this research are either from ALTM Pegasus or ALTM Gemini only, since ALTM Aquarius has just been added lately to the fleet of DAC's airborne laser scanners. Also, as its name implies, ALTM Aquarius is designed for LiDAR mapping of coastal areas and bodies of water, which is not the focus of this research. ALTM Pegasus has a laser pulse rate of 100,000 up to 500,000 points per second (Optech, 2010). On the other hand, ALTM Gemini has a lower laser pulse rate of 33,000 up to 167,000 points per second (Optech, 2009). Therefore, LAS file generated using ALTM Pegasus is generally denser than to those of ALTM Gemini's.

The input files used in this research are in LAS file format with version 1.2. Version 1.2 is chosen in compliance to the latest LAS version recognized by TerraScan, the software used by the DPC for their data processing workflow. Each LAS file covers an area of 1 km * 1km block, located usually over flood plain areas which are the primary interest of UP DREAM. The flood plain areas covered by the sample data are characterized by flat slope and has a land use for residential, agricultural and several commercial purposes. The input LAS files are preprocessed and have undergone positional accuracy assessment using the software LiDAR Mapping Suite (LMS).

A total of twenty (20) LAS files were initially requested from UP DREAM to be used as samples for this research. However, it is noted that requesting twenty (20) LAS files might be difficult for approval due to the sensitivity and acquisition cost of the LAS files. In the end, twelve (12) LAS files are provided by UP DREAM and are used in this research. An "End User License Agreement" is signed which signifies that the LAS files provided may only be used solely for the accomplishment of this research. The 12 LAS files are chosen randomly by accessing the archives of pre-processed LAS tiles and randomly opening files. Files are visually inspected and once non-terrestrial points are present, the opened file is a candidate for sample data.

Table 3.1 The 12 sample LiDAR data used

LAS file	Number of LiDAR points	ALTM sensor used
Agno5M_068	4,860,042	Pegasus
Agno5M_115	5,623,718	Pegasus
Cagayan21D_012	1,830,950	Gemini
Cagayan21D_020	405,310	Gemini
Cagayan21D_046	870,872	Gemini
Cagayan21D_022	2,430,627	Gemini
Pampanga3B_009	3,545,823	Pegasus
Pampanga3B_108	3,122,311	Pegasus
Pampanga3B_165	3,032,723	Pegasus
Pampanga3B_102	3,740,053	Pegasus
Pampanga8I_127	3,322,099	Gemini
Pampanga8I_112	3,228,089	Gemini

3.2 Preparation of Python Script for Automated Filtering

Python 2.7 is first installed on the computer to be able to create a script for automation. Then additional libraries Numpy and Laspy are also installed. Laspy can be downloaded for free in this URL: https://pypi.python.org/packages/any/l/laspy/laspy-1.2.5.win32.exe.

3.3 Automated Filtering Algorithm

The filtering algorithm starts by opening the LAS file by using `laspy.File` object. Then, the X, Y and Z coordinates of all the LiDAR points saved in the LAS file are arranged into a single two-dimensional Numpy array. Each row contains the coordinates of a point. Therefore, the number of rows is equal to the number of points in the input LiDAR data. The number of columns on the other hand is equal to three (3), each column representing the set of coordinates of points along each of the three (3) orthogonal axis in the 3D space.

A histogram of points will be created. The histogram will be created based on the elevations of the points. The bins of the histogram will start from the elevation of the lowest point in the point cloud and will end on the elevation of the lowest point plus one thousand (1,000) meters. The optimal interval for the elevation is not determined at first. In the example in Table 3, 10 m is the interval used. After successfully creating the elevation histogram, the script will search for the very first bin where the population of points is zero. In the example, the very first bin with the zero population of points is 136.32 m – 146.32 m. It is observed that the some bins with higher elevation to 136.32 m – 146.32 m has a population of points that is not empty. It

is suspicious that there is not a single LiDAR point lying between the elevation of 136.32 m and 146.32 m, a 10 m vertical distance, while above 146.32 m, LiDAR points are present. Therefore LiDAR points above 146.32 m are "disconnected" from the points below 136.32 m with a distance of at least 10 m. Points above 146.32 can be considered to be "floating" and therefore can be considered as non-terrestrial points. Therefore, points above 146.32 m (with red font on the table) will be deleted by the Python script while points below 136.32 will be saved.

Table 3.2 Elevation Histogram of Agno5M_068 with 10 m as elevation interval

Elevation range (bins)	Number of points with the interval
36.32 m – 46.32 m	153
46.32 m – 56.32 m	4,686,455
56.32 m – 66.32 m	172,028
66.32 m – 76.32 m	1,306
76.32 m – 86.32 m	5
. . .	. . .
116.32 m – 126.32 m	4
126.32 m – 136.32 m	2
136.32 m – 146.32 m	0
146.32 m – 156.32 m	2
156.32 m – 166.32 m	1
. . .	. . .
1036.22 m – 1036.32 m	0

This simple concept of non-terrestrial point filtering takes approximately 3 seconds (using the same computer used by UP DREAM) compared to manual filtering which takes approximately a minute. However, one problem is that some non-terrestrial points may be isolated and low – lying (e.g. point caused by a bird flying along an urban area with tall

buildings) and generating a single elevation histogram for the entire LAS file to filter the non-terrestrial points may not be effective. A solution to this problem is to segment the LiDAR point cloud along the X axis and create an elevation histogram for each segment that are independent

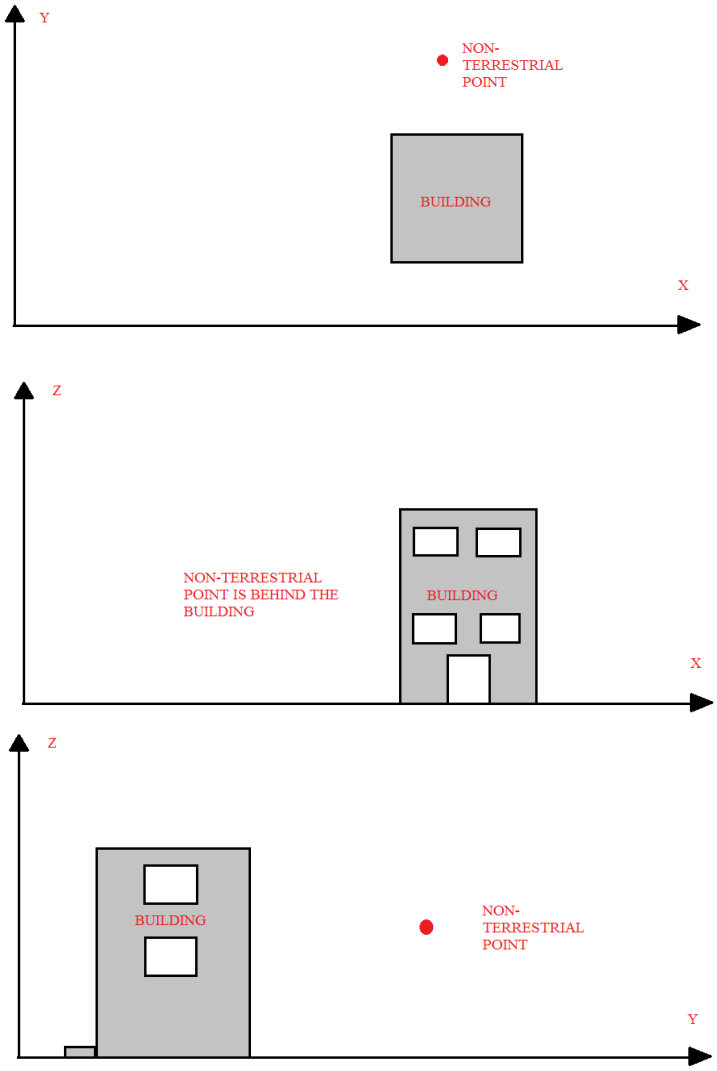

Figure 3.1 Diagram showing the possibility of not deleting a non-terrestrial point because of using only a single elevation histogram for the entire point cloud

from each other. Same thing must be done along the Y axis. If a certain point is flagged as a non-terrestrial point based on the X-axis segmentation but not flagged as a non-terrestrial point based on Y-axis segmentation or vice versa, then that point is considered as a non-terrestrial point. This

method minimizes the error caused by an isolated non-terrestrial point that is mixed with the populated bin in the elevation histogram. One hundred (100) meters is the suggested interval for segmenting the point cloud because using using smaller interval increases the running time of the algorithm.

Here is the pseudocode of the Python script used to automatically filter non-terrestrial points:

Import Laspy and Numpy
Open LAS file using `laspy.File`
Convert point data record of the Laspy into a Numpy Array using `numpy.vstack`

Segment Numpy array along X axis using `numpy.logical_and`
For every segment in all segments along X axis:
    Create elevation histogram for each segment using `numpy.histogram`
    Find lowest bin with empty points
    Flag all points above bin as non-terrestrial

Segment Numpy array along Y axis:
For every segment in all segments along Y axis:
    Create elevation histogram for each segment
    Find lowest bin with empty points
    Flag all points above bin as non-terrestrial

For every point in the LAS file:
    If point is flagged as non-terrestrial at least once: (using `numpy.logical_or`)
        Point is non-terrestrial
    Else:
        Point is terrestrial

Write a new LAS file with only terrestrial points using `numpy.File`

The following is the actual Python script used:

```
from laspy.file import File
import numpy, datetime, math

#benchmark
start = datetime.datetime.now()
```

```
#######ENTER PARAMETERS HERE#################
filename = "cag21d_022"
elev_increment = 9
min_points = 0
xy_increment = 1

###########ELEV HISTOGRAM FUNCTION##########
def elev_hist(elev_array, incre):
    lowest = numpy.min(elev_array)
    incre_high = numpy.max(numpy.diff(numpy.sort(elev_array)))
    incre_low = numpy.diff(numpy.sort(elev_array))

    amount, limit = numpy.histogram(elev_array, bins =
     numpy.arange(lowest, lowest + 1000, incre))

    a = 0
    max_elev = numpy.max(elev_array)
    for x in amount:
        if x <= min_points:
            max_elev = limit[a]
            break
        a += 1

    return max_elev, amount[a:].sum()

###### END OF ELEV HISTOGRAM FUNCTION ##########

inFile = "./inputs/" + filename + ".las"
inFile = File(inFile, mode="r")
coords = numpy.vstack((inFile.x, inFile.y, inFile.z,
numpy.arange(len(inFile.Z)))).transpose()
lowx = (math.floor(inFile.header.min[0]/1000)*1000)
lowy = (math.floor(inFile.header.min[1]/1000)*1000)
x_axis = numpy.linspace(lowx, lowx+1000, num=xy_increment+1)
y_axis = numpy.linspace(lowy, lowy+1000, num=xy_increment+1)

total_deleted_x = 0
i = 0
while i <xy_increment:
    if i==0:
        x_strip = coords[numpy.logical_and((coords[:,0] >=
          x_axis[i]), (coords[:,0] <=x_axis[i+1]))]
        y_strip = coords[numpy.logical_and((coords[:,1] >=
          y_axis[i]), (coords[:,1] <=y_axis[i+1]))]
    else:
        x_strip = coords[numpy.logical_and((coords[:,0] >
          x_axis[i]), (coords[:,0] <=x_axis[i+1]))]
```

```
        y_strip = coords[numpy.logical_and((coords[:,1] >
          y_axis[i]), (coords[:,1] <=y_axis[i+1]))]
    x_bound, x_deleted_pts = elev_hist(x_strip[:,2],
     elev_increment)
    y_bound, y_deleted_pts = elev_hist(y_strip[:,2],
     elev_increment)
    x_safe = numpy.vstack(((x_strip[:,2] < x_bound),
     x_strip[:,3])).transpose()
    y_safe = numpy.vstack(((y_strip[:,2] < y_bound),
     y_strip[:,3])).transpose()
    if i==0:
        clean_x = x_safe
        clean_y = y_safe
    else:
        clean_x = numpy.vstack((clean_x, x_safe))
        clean_y = numpy.vstack((clean_y, y_safe))
    i += 1

clean_x = clean_x[clean_x[:,1].argsort()]
clean_y = clean_y[clean_y[:,1].argsort()]
cleanliness = numpy.logical_and(clean_x[:,0], clean_y[:,0])

output_file = "./outputs/clean_" + filename + "legit.las"
output_file = File(output_file, mode = "w", header =
inFile.header)
output_file.points = inFile.points[cleanliness]

output_file.close()
inFile.close()

duration = datetime.datetime.now() - start
print "run time: " + str(duration)
```

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Accuracy of the Automation and Selection of Optimal Elevation Interval

The created Python script is applied to all the twelve (12) sample LAS files. Different elevation intervals are tried to empirically determine the optimal elevation interval for the twelve (12) LAS files. A table is created (Table 4.1) to compare the number of points deleted by the script against the number of points deleted manually.

Table 4.1 Number of points deleted manually vs points deleted by the Python script

		ELEVATION INTERVAL								
	manual	6m	7m	8m	9m	10m	11m	12m	13m	14m
Agn5m 068	97	98	98	95	95	91	93	87	86	90
Agn5m 155	41	43	42	41	42	39	40	36	35	33
cag21d_012	217005	217005	217005	217005	217005	217005	217005	217005	217005	217005
cag21d_020	354561	354561	354561	354561	354561	354561	354561	354561	354561	354561
cag21d_046	331880	331880	331880	331880	331880	331880	331880	331880	331880	331880
cag21d_022	356237	356237	356237	356237	356237	356237	356237	356237	356237	356237
pam3b_009	53	56	54	53	53	53	53	53	53	53
pam3b_108	51	58	57	56	57	54	49	51	46	45
pam3b_165	1376	567420	787	788	784	784	779	781	781	769
pam3b_102	41	40	41	41	41	40	40	41	41	40
pam8i_127	60	63	63	60	60	40	40	60	39	40
pam8i_112	42	53	53	43	43	43	43	43	42	43

According to Table 4.1, the cells highlighted in red means that the number of points deleted by the script is greater than the number of points deleted manually. The green cells states that points deleted by the script is lesser. The yellow cells state that the points deleted by the script is equal to the points deleted manually. Based on the table, there is a trend showing that the number of points deleted by the script increases if elevation interval decreases and vice versa. The probable optimal value for the elevation interval is between 6 m and 14 m. Decreasing the elevation interval lower than 6 m might cause additional increase in the number of automatically deleted points thus moving away from the suggested number of points that must be deleted. On

the other hand, increasing the elevation interval above 14 m may lessen the number of automatically deleted points thus also moving away from the suggested number of points to be deleted. It is also noticeable that the all the sample data from Cagayan21D have been automatically filtered perfectly from non-terrestrial points. It is due to the fact that the non-terrestrial points present in LAS files of Cagayan21D are caused by clouds and not by isolated airborne objects. Non-terrestrial points caused by clouds are usually characterized by being clustered and have large vertical distance from terrestrial points.

Table 4.2 Errors in automatic filtering (excess or lacking points)

	6m	7m	8m	9m	10m	11m	12m	13m	14m
Agn5m 068	1	1	2	2	6	4	10	11	7
Agn5m 155	2	1	0	1	2	1	5	6	8
cag21d_012	0	0	0	0	0	0	0	0	0
cag21d_020	0	0	0	0	0	0	0	0	0
cag21d_046	0	0	0	0	0	0	0	0	0
cag21d_022	0	0	0	0	0	0	0	0	0
pam3b_009	3	1	0	0	0	0	0	0	0
pam3b_108	7	6	5	6	3	2	0	5	6
pam3b_165	566044	589	588	592	592	597	595	595	607
pam3b_102	1	0	0	0	1	1	0	0	1
pam8i_127	3	3	0	0	20	20	0	21	20
pam8i_112	11	11	1	1	1	1	1	0	1

It is shown in table 4.2 that the number of points deleted using the automatic filtering of Pampanga3B_165 is too far from the manual filtering. The raw (unfiltered version) of Pampanga3B_165 is visually inspected and is shown in Figure 4.1. Based on the inspection of point data record of non-terrestrial points in Pampanga 3B_165, it is shown that these non-terrestrial points have extremely low intensity values (0 – 10) and therefore do not represent physical objects.

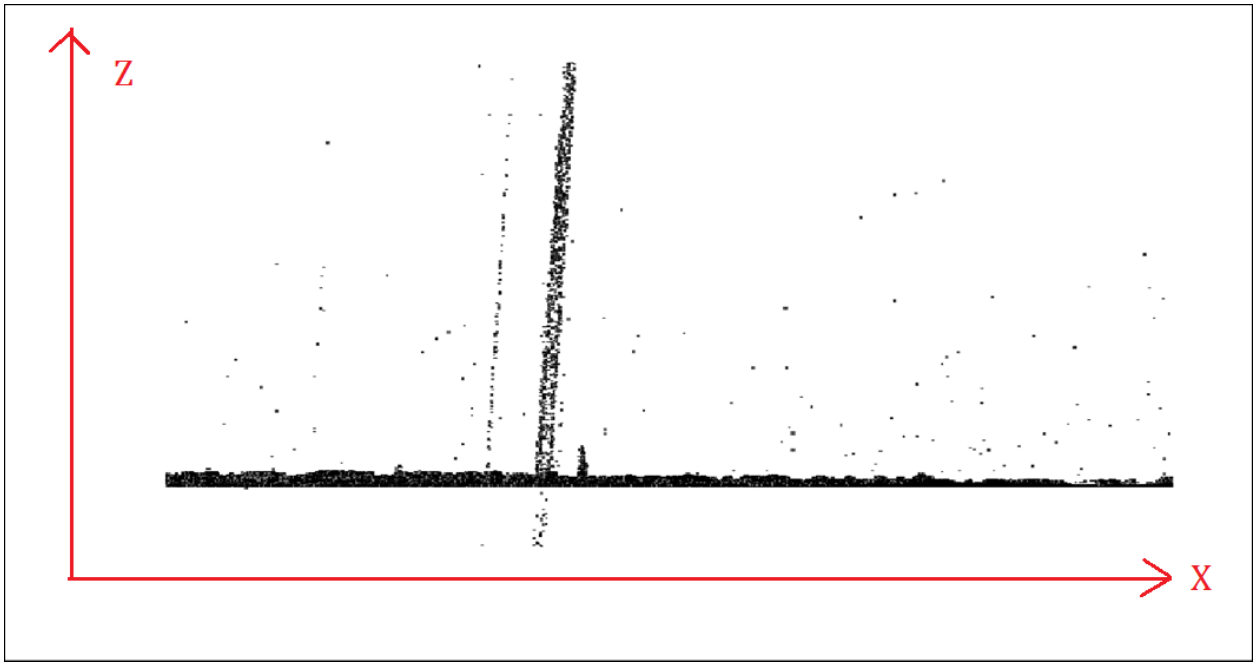

Figure 4.1 Cross-sectional view of Pampanga3B_165 point clouds

A seperate Python script is created for the Pampanga3B_165 to remove points with extremely low intensity. Using the other eleven (11) sample data, it is shown that using an elevation interval of 8 m is the optimal value for the sample dataset. The accuracy of each elevation interval on each sample data is computed using the following formula: accuracy = (auto - |auto – manual|) * 100 / manual

Table 4.3 Accuracy of Automated Filtering using different elevation intervals

	6m	7m	8m	9m	10m	11m	12m	13m	14m
Agn5m 068	98.9690722	98.9690722	97.9381443	97.9381443	93.814433	95.8762887	89.6907216	88.6597938	92.7835052
Agn5m 155	95.1219512	97.5609756	100	97.5609756	95.1219512	97.5609756	87.804878	85.3658537	80.4878049
cag21d_012	100	100	100	100	100	100	100	100	100
cag21d_020	100	100	100	100	100	100	100	100	100
cag21d_046	100	100	100	100	100	100	100	100	100
cag21d_022	100	100	100	100	100	100	100	100	100
pam3b_009	94.3396226	98.1132075	100	100	100	100	100	100	100
pam3b_108	86.2745098	88.2352941	90.1960784	88.2352941	94.1176471	96.0784314	100	90.1960784	88.2352941
pam3b_102	97.5609756	100	100	100	97.5609756	97.5609756	100	100	97.5609756
pam8i_127	95	95	100	100	66.6666667	66.6666667	100	65	66.6666667
pam8i_112	73.8095238	73.8095238	97.6190476	97.6190476	97.6190476	97.6190476	97.6190476	100	97.6190476
	94.6432414	95.6080067	98.7048428	98.3048602	94.9909747	95.5783987	97.7376952	93.5656114	93.0321176

In can be seen that the optimal value for the elevation interval is equal to eight (8) meters which has the highest accuracy of 98.70% for the eleven (11) LiDAR data. Using the filtering algorithm with 8-meter elevation interval, non-terrestrial points from the sample data are effectively removed.

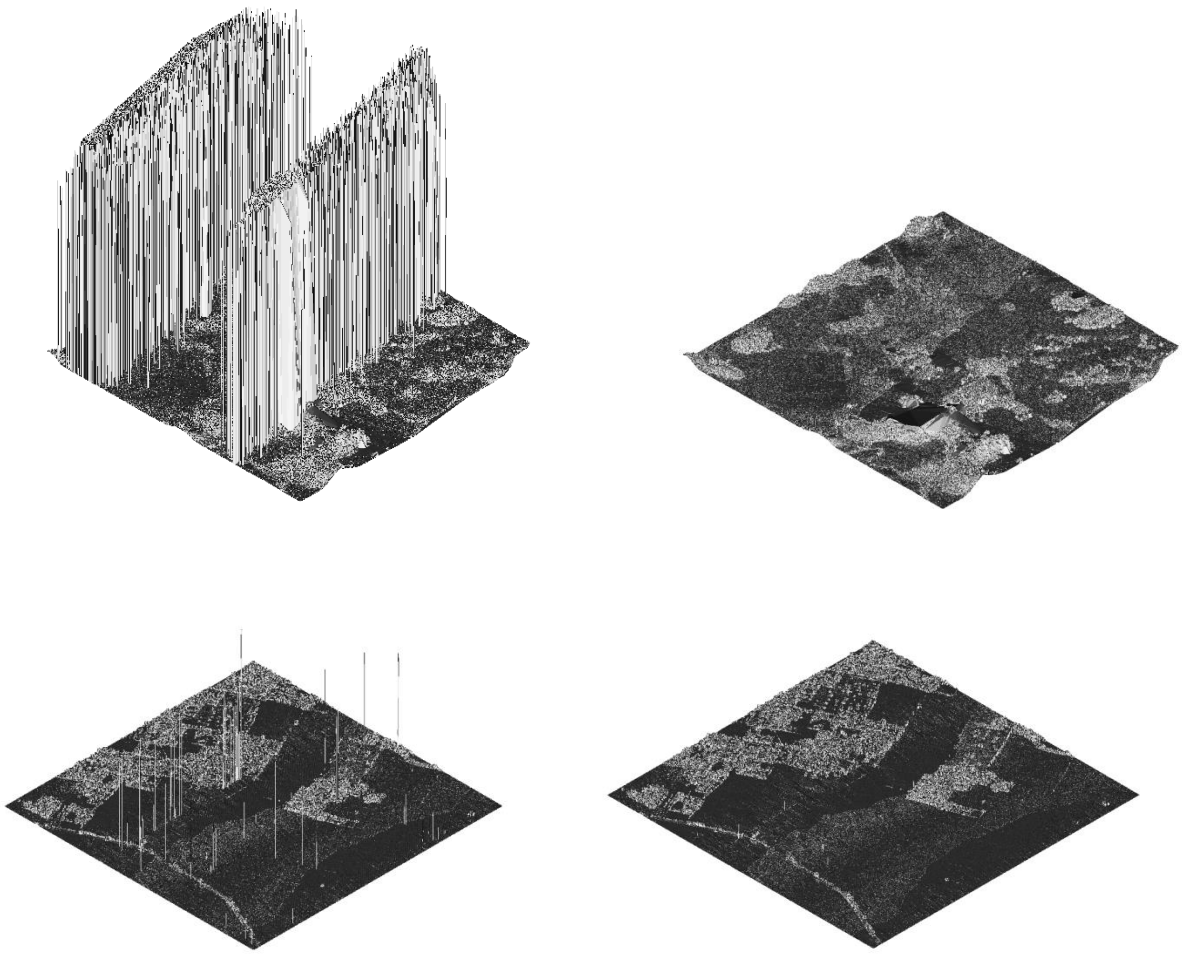

Figure 1 DEMs of raw and filtered data. Upper Left : raw Cag21D. Upper Right: filtered Cag21D. Lower Left: raw Agn5M. Lower Right: filtered Agn5M

4.2 Applicability of the Automated Filtering Algorithm in the DPC Workflow

The number sample data used in this research is not adequate to make the automated filtering algorithm to be robust. Additional sample data cannot be requested. However, the algorithm has performed well for the eleven out of twelve sample data. One significant advantage of using the automated filtering algorithm is the duration it takes to filter the non-terrestrial points.

4.3 Limitations of the Automated Filtering Algorithm

The algorithm cannot “intelligently” distinguish non-terrestrial point from those that are terrestrial because it cannot exactly specify what object a point represents. The only basis that the algorithm uses are from the items in the point data record which are the Cartesian coordinates and the intensity. However, even in manual filtering, the basis of the personnel for classifying non-terrestrial points are the Cartesian coordinates and the morphology of the points. The positions and intensity of the points are adequate to identify non-terrestrial points and specifying what object every point represent is unnecessary. It would also be difficult for the algorithm to distinguish non-terrestrial points that are spatially near or mix with terrestrial points. Doing it manually will just be as difficult.

CHAPTER 5

CONCLUSION

The automated filtering algorithm created has effectively filtered non-terrestrial points from the eleven (11) LAS files from UP DREAM. Applying the optimal elevation interval of eight (8) meters yields an accuracy of up to 98.70%. However this elevation interval of eight (8) meters is optimal only for the sample data. It is not conclusive that the algorithm will provide optimal solutions for the other LAS files of UP DREAM. Large number of sample data can optimize the algorithm but access to these large number of LAS files from UP DREAM is restricted due to the acquisition cost and data sensitivity. However, this is a good start for automating the filtering process done by DPC. Using the automated filtering algorithm can save a significant amount of time from the DPC processing workflow.

References:

Baltsavias, E., 1999, A comparison between photogrammetry and laser scanning. ISPRS J. Photogramm. Remote Sens. 1999, 54, pp 83-94

Brown, G., Butler, H., 2012. Laspy: Documentation. laspy.readthedocs.org

Li, Y., Wu, H., Xu, H., An, R., Xu, J., He, Q., 2013 A gradient- constrained morphological filtering algorithm for airborne LiDAR. Optics & Laser Technology. pp 288-296

Liu, X., 2008, Airborne LiDAR for DEM generation: some critical issues. Prog. Phys. Geog. pp 31-49

Mallet, C., Bretar, F., Roux, M., Soergel, U., Heipke, C., 2011. Relevance assessment of full-waveform lidar data for urban area classification. ISPRS Journal of Photogrammetry and Remote Sensing 66. pp 71-84

Meng, X., Currit, N., Zhao, K., 2010, Ground Filtering Algorithms for Airborne LiDAR Data: A Review of Critical Issues. Remote Sensing

Mongus, D., Lukac, N., Zalik, B., Ground and building extraction from LiDAR data based on differential morphological profiles and locally fitted surfaces. ISPRS Journal of Photogrammetry and Remote Sensing

Optech Incorporated, 2009. Gemini Summary Specification Sheet. www.optech.com

Optech Incorporated, 2010. Pegasus Summary Specification Sheet. www.optech.com

Schmid, K., et. al. 2008. LiDAR 101: An Introduction to LiDAR Technology, Data, and Applications. National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center. pp 1-38

The American Society for Photogrammetry & Remote Sensing 2008. LAS Specification Version 1.2. The American Society for Photogrammetry & Remote Sensing. pp 2-13.

The American Society for Photogrammetry & Remote Sensing., 2013. LAS Certification Version 1.4 – R13. The American Society for Photogrammetry & Remote Sensing. pp 2.

Watershed Sciences, 2006. LiDAR Remote Sensing Data Collection: Desolation Creek, Middle Fork John Day River, & John Day River, Oregon. Puget Sound LiDAR Consortum.