

BorderBlaze

**An Automated Tool for Boundary Point Segmentation from Aerial LiDAR Point
Cloud Data**

Submitted by

Mushfiquur Rahman Chowdhury

BSSE 1110

Supervised by

Dr. Emon Kumar Dey

Associate Professor

Institute of Information Technology

University of Dhaka



Submission Date: 17 December 2023

Signature Page

Project: BorderBlaze

Author: Mushfiqur Rahman Chowdhury

Submitted: 17.12.23

Supervised By:

Dr. Emon Kumar Dey,

Associate Professor,

Institute of Information Technology,

University of Dhaka

Supervisor's Approval:

Letter of Transmittal

17 December 2023

BSSE 4th Year Exam Committee

Institute of Information Technology

University of Dhaka

Subject: Submission of the final report on **BorderBlaze: An Automated Tool for Boundary Point Segmentation from Aerial LiDAR Point Cloud Data**

Sir,

With due respect, I am pleased to submit the final report of Software Project Lab 3 on “**BorderBlaze: An Automated Tool for Boundary Point Segmentation from Aerial LiDAR Point Cloud Data**”. I have tried my best to deliver a good understanding of the details of the project. However, it might lack perfection. So, I, therefore, hope that you would be kind enough to accept my report and oblige thereby.

Sincerely yours,

Mushfiqur Rahman Chowdhury

Roll: BSSE 1110

Institute of Information Technology

University of Dhaka

Acknowledgment

I am grateful to the Institute of Information Technology for giving me such a tremendous opportunity to work on “BorderBlaze - Automated Tool for Boundary Point Segmentation from Aerial LiDAR Point Cloud Datasets”. I want to convey my deep and sincere gratitude to my supervisor, Dr. Emon Kumar Dey, Associate Professor, Institute of Information Technology, University of Dhaka, for providing guidelines throughout this project and in the preparation of this analysis document. His dynamism, vision, sincerity, and motivation have deeply inspired me. Lastly, I would like to thank my classmates. They have always been helpful and provided valuable insights from time to time.

Abstract

BorderBlaze is an automated boundary point segmentation tool for aerial LiDAR point cloud data. The document contains its software requirements and specifications, architectural design, implementation details, user manual, and testing. Aerial LiDAR (Light Detection and Ranging) point cloud data refers to a collection of three-dimensional coordinates generated by a LiDAR system mounted on an aerial platform such as an aircraft, drone, or satellite. The tool works on the LiDAR point cloud data of building roofs from a site of Vaihingen. The boundary points of the roof LiDAR point cloud data are segmented by the tool using three different methods Neighborhood Approach, Concave Hull Algorithm, and Delaunay Triangulation. The segmented boundary points of the roof properly represent the outline of the area occupied by the building. The outline can eventually be applied to measurements and different areas of urban planning

Table of Content

Signature Page	2
Letter of Transmittal	3
Acknowledgment	4
Abstract	5
Table of Content	6
List of Figures	8
1. Introduction	9
1.1 Purpose	9
1.2 Scope	9
1.3 Assumptions	10
1.4 Motivation	10
2. Project Description	11
2.1. Quality Function Deployment	11
3. Scenario Based Modeling	13
3.1. Use Case Diagram	13
3.1.1. Point Cloud Boundary Segmentation Tool	13
3.1.2. Modules of Point Cloud Boundary Segmentation Tool	13
3.1.2.1. Input Data Processor	14
3.1.2.2. Data Plot Generator	15
3.1.2.3. Boundary Detector	15
3.1.2.3. Result Display	15
4. Class Based Modeling	16
4.1. Analysis Classes	16
4.2. Class Cards	16
4.3. Class Diagram	20
5. Architectural Design	21
5.1. Architectural Context Diagram	21
	6

5.2. Defining Archetypes	21
5.3. Refining the Architecture into Top-Level Components	22
5.4. Mapping Requirements to Software Architecture	23
6. Methodology	25
6.1. Workflow of BorderBlaze	25
6.2. Three-Dimensional Plotting of LiDAR Point Cloud Data	25
6.3. Boundary Point Segmentation	26
6.3.1. Neighborhood Based Approach	26
6.3.2. Concave Hull Algorithm Approach	27
6.3.3. Delaunay Triangulation Approach	30
6.4. Vaihingen Dataset	33
7. User Interface Design	34
7.1 User Analysis	34
7.2. Task Analysis	35
7.3. User Interface and User Manual	36
7.3.1. BorderBlaze Home Page:	36
7.3.2. View Neighborhood Based Approach Result	38
7.3.3. View Concave Hull Algorithm Result	39
7.3.4. View Delaunay Triangulation Result	41
8. Testing	43
8.1. High-Level Description of Testing Goals	43
8.2. Test Cases	44
8.3 Item Pass/Fail Criteria	44
9. Conclusion	46
References	47

List of Figures

Figure 1: Level 0 Use Case.....	13
Figure 2: Level 1 Use Case.....	14
Figure 3: Class Diagram.....	20
Figure 4: Defining Archetypes.....	22
Figure 5: Refining Archetypes to Components.....	23
Figure 6: Mapping Requirements to Software Architecture	24
Figure 7: Workflow of BorderBlaze.....	25
Figure 8: KNN Pseudocode.....	27
Figure 9: Convex Hull vs Concave Hull Algorithm	28
Figure 10: Modified Concave Hull Algorithm.....	29
Figure 11: Thresholding based on edge length.....	30
Figure 12: Delaunay Triangulation.....	31
Figure 13: Delaunay Triangulation Pseudocode	32
Figure 14: Individual Building Extraction from Vaihingen Dataset.....	33
Figure 15: BorderBlaze Home Page	37
Figure 16: View Input 3D Plotting.....	38
Figure 17: Submit Neighborhood Based Approach	38
Figure 18: Neighborhood Approach Results.....	39
Figure 19: Submit Concave Hull Algorithm	40
Figure 20: Concave Hull Algorithm Results.....	40
Figure 21: Submit Delaunay Triangulation Method	41
Figure 22: Delaunay Triangulation Method Results	42

1. Introduction

A point cloud refers to a collection of data points that are defined by their coordinates in a specific coordinate system. These data points are typically used to represent the external surface of an object in three-dimensional space. The boundary of a 3D point cloud typically involves extracting a surface or enclosing shape that encapsulates the points in the cloud. Their accurate segmentation is crucial for various applications, such as object segmentation, scene understanding, and obstacle detection.

1.1 Purpose

The purposes of this document are-

- Identify and analyze the requirements
- Design the architecture
- Design the test plan
- Describe methodology
- Reduce the development effort
- Improve understanding

1.2 Scope

The scope of the project is given below-

- Web Application tool
- Three methods for boundary segmentation in point cloud dataset will be used in the project:
 - Neighborhood based approach

- Concave Hull Algorithm
- Triangulation-based technique

1.3 Assumptions

I am assuming that the project will prove to be more useful and provide better results when the following criterias are met by the dataset-

- The point cloud dataset represents a real-life scene containing various objects.
- Each point in the point cloud dataset will contain at least the value of x and y coordinates.

1.4 Motivation

Point cloud boundary point segmentation is crucial to recognize and segment objects from a point cloud dataset. It is valuable in applications like autonomous vehicles, robotics and augmented reality where understanding the boundaries of objects is essential for decision-making and interaction.

The established boundary point segmentation methods defined for images cannot be directly applied to point cloud data. One of the reasons is that the data representation is different. While an image can be structured as a matrix, a point cloud is an unorganized and irregularly distributed set of scattered points. Secondly, the neighborhood of a point in point cloud is more complex and cannot be arranged in a grid-like pattern of an image pixel.

2. Project Description

Point cloud data is used in a wide range of applications across various industries. It can provide a highly accurate representation of the real world in three dimensions. The segmentation of boundary points in a point cloud data involve different complicated methods. There is no one-size-fits-all "best" method to detect boundaries in point cloud data, as the choice of method depends on various factors, including the characteristics of the data, the specific application, and the desired level of accuracy. Different methods have their strengths and weaknesses, and the most suitable method can vary from one scenario to another.

The proposed software tool addresses the problem by letting the user detect boundary points of the point cloud data using one to three methods and choose the best output based on the variables.

2.1. Quality Function Deployment

Quality function deployment (a model for product development and production) is a focused methodology for carefully listening to the customers or consumers of that particular product and then effectively responding to those identified needs and expectations.

With respect to this project, the following requirements are identified-

- **Normal Requirements**

NR1 - Web application tool

NR2 - Users can input point cloud data by Direct File Upload.

NR3 - Users can choose one or more methods to detect boundary points

- Neighborhood based approach
- Concave Hull Algorithm
- Triangulation-based technique

NR4 - Boundary points detected using one to three of the following methods: Neighborhood approach, Concave Hull Algorithm, Delaunay Triangulation.

NR5 - The input point cloud data will be plotted and shown in the user interface as a representation of the input.

NR6 - The output point cloud data (boundary highlighted) will be plotted and shown in the user interface .

- **Expected Requirements**

ER1 - The user interface will contain buttons to choose among the methods to use for detecting boundary points from the data.

ER2 - Input file or folder from local file system

- **Exciting Requirements**

ExcR1 - The input and output plottings of the point cloud data will be shown together so that the boundary points can be easily visualized.

ExcR2 - Users can upload point cloud data of different formats such as ply, xyz.

3. Scenario Based Modeling

3.1. Use Case Diagram

A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled. Use cases specify the expected behavior (what), and not the exact method of making it happen (how).

3.1.1. Point Cloud Boundary Segmentation Tool

Primary actor: User

Secondary actor: No secondary actor

Level 0

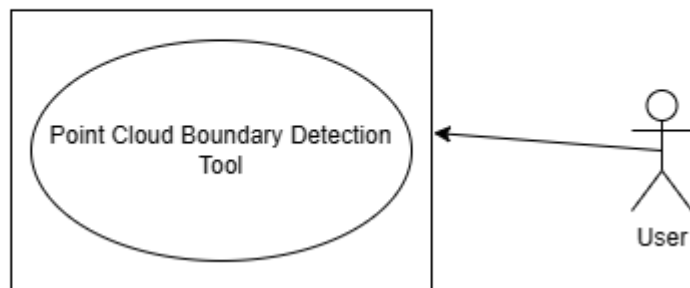


Figure 1: Level 0 Use Case

3.1.2. Modules of Point Cloud Boundary Segmentation Tool

Primary actors: User

Secondary actors: No secondary actor

Level 1

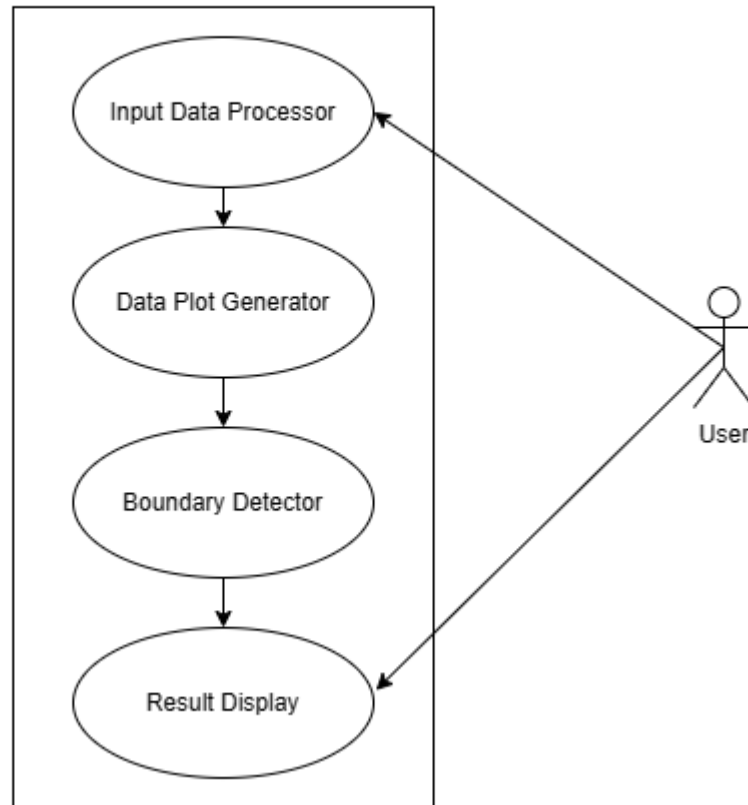


Figure 2: Level 1 Use Case

3.1.2.1. Input Data Processor

The module is responsible for receiving input point cloud data from the user. Users can input point cloud data in three ways: Direct file upload, drag and drop, clipboard copy-paste. This module will read the dataset using necessary libraries and structure them into a dataframe or similar format form which we will be able to plot the point cloud data or process it further for boundary point segmentation. If the provided data is too large, the module will try to compress it or let the user know about its compatibility. This module will also receive input on the methods

to be used for boundary detection in the point cloud data. The user may choose between one to three methods to be used for boundary point segmentation.

3.1.2.2. Data Plot Generator

The module plots the point cloud dataset so that the user will be able to visualize the input and output data in a diagram. The data plot will be shown in the user interface and also at the same time the output data in which the boundary has been highlighted can be generated in different file formats such as xyz, ply or png.

3.1.2.3. Boundary Detector

The boundary detector module will use the processed point cloud data to detect boundary points from it. Boundary points will be detected in the following methods: Neighborhood based approach, Concave Hull Algorithm, Triangulation based technique.

3.1.2.3. Result Display

The result of the boundary detected points will be highlighted to a particular color or unique intensity level and plotted data will be shown in the user interface. The output data can be downloaded by the user in different formats such as xyz, ply or png.

4. Class Based Modeling

4.1. Analysis Classes

After identifying nouns from the scenario, I have filtered nouns belonging to the solution domain using General Classification (External entities, Things, Events, Roles, Organizational units, Places, and Structures). Nouns selected as a potential class were filtered using Selection Criteria (Retained information, Needed services, Multiple attributes, Common attributes, Common operations, and Essential requirements). After performing analysis on potential classes, I have found the following analysis classes-

1. InputDataProcessor
2. DataPlotGenerator
3. UI
4. BoundaryDetector
5. OutputDataHandler
6. UIController

4.2. Class Cards

InputDataProcessor	
Attributes	Methods

- pointCloudData - boundarySegmentationMethods[]	+ readPointCloudData() - preprocessPointCloudData() + getPointCloudData() + setPointCloudData() + addMethods() + plotInputData()
Responsibility	Collaborator
<ul style="list-style-type: none"> ● Handle User Input ● Preprocess the point cloud data 	<ul style="list-style-type: none"> ● UIController ● DataPlotGenerator

Table 1: Class card of InputDataProcessor

DataPlotGenerator	
Attributes	Methods
- pointCloudDataFrame	+ plotPointCloudData() + displayDiagram()
Responsibility	Collaborator
<ul style="list-style-type: none"> ● Generates plot of point cloud data for visualization. 	<ul style="list-style-type: none"> ● UI ● InputDataHandler ● OutputDataHandler

Table 2: Class card of DataPlotGenerator

UI	
Attributes	Methods
<ul style="list-style-type: none"> - UIMainWindow - Widgets 	<ul style="list-style-type: none"> + setupUI() + getWidget()
Responsibility	Collaborator
<ul style="list-style-type: none"> • Manage the graphical user interface. 	<ul style="list-style-type: none"> • UIController • DataPlotGenerator

Table 3: Class card of UI

UIController	
Attributes	Methods
<ul style="list-style-type: none"> - inputData - outputData - UI 	<ul style="list-style-type: none"> + uiController() + eventHandler() + updateUI()
Responsibility	Collaborator
<ul style="list-style-type: none"> • Controls user interaction with UI. 	<ul style="list-style-type: none"> • InputDataProcessor • OutputDataHandler

Table 4: Class card of UIController

BoundarySegmentor

Attributes	Methods
<ul style="list-style-type: none"> - processedPointCloudData - highlightedBoundaryPointsData 	<ul style="list-style-type: none"> + segmentBoundary() - neighborhoodBoundarySegmentationMethod() - concaveHullAlgorithm() - delaunayTriangulation() + highlightBoundaryPoints()
Responsibility	Collaborator
<ul style="list-style-type: none"> • Detects boundary points from point cloud data. • Applies boundary point segmentation methods and algorithms 	<ul style="list-style-type: none"> • InputDataProcessor • OutputDataHandler

Table 5: Class card of BoundaryDetector

OutputDataHandler	
Attributes	Methods
<ul style="list-style-type: none"> - outputPointCloudData 	<ul style="list-style-type: none"> + postProcessPointCloudData() + saveOutputData() + generateXYZFile() + generatePLYFile() + generatePNGFile()
Responsibility	Collaborator

<ul style="list-style-type: none"> • Handle the final output data after boundary points are detected 	<ul style="list-style-type: none"> • UIController • DataPlotGenerator
---	---

Table 6: Class card of OutputDataHandler

4.3. Class Diagram

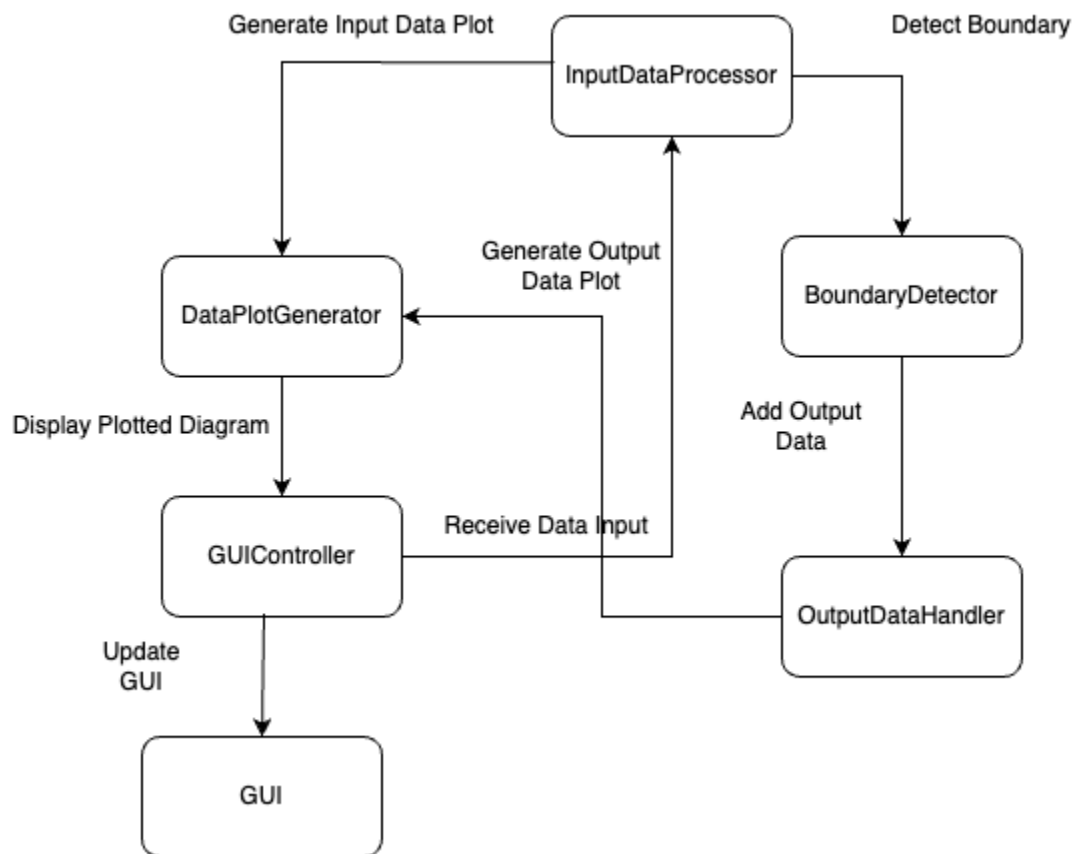


Figure 3: Class Diagram

5. Architectural Design

5.1. Architectural Context Diagram

It is possible to model how software communicates with entities outside of its boundaries using an architectural context diagram. The target system's cooperating systems are denoted by the following symbols:

Superordinate systems: Those systems that use the target system as part of some higher-level processing scheme. The proposed project has no superordinate systems.

Subordinate systems: Those systems that are used by the target system and provide data or processing that are necessary to complete the target system functionality. There are no subordinate systems in our project.

Peer-level systems: Those systems that interact on a peer-to-peer basis (i.e., information is either produced or consumed by the peers and the target system. There are no peer-level systems in our project.

Actors: Entities (people, devices) that interact with the target system by producing or consuming information that is necessary for requisite processing. The user of our system is the Actor in this case. Each of these external entities communicates with the target system through an interface (the small shaded rectangles).

5.2. Defining Archetypes

An archetype refers to a class or template representing a fundamental concept crucial for shaping the architecture of the intended system. In essence, even highly intricate systems can be devised by employing a relatively limited set of these archetypes. These archetypes represent the enduring elements of the desired system structure, although they can take on different forms

depending on how the system behaves. Together, these archetypes constitute the architecture of the intended system.

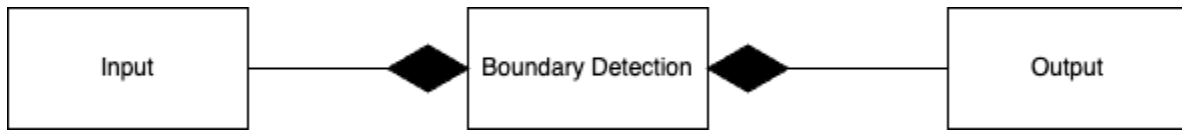


Figure 4: Defining Archetypes

Boundary Detection: An abstraction that depicts all the mechanisms of the tool to detect boundary points of the point cloud data.

Input: An abstraction that depicts all the input mechanisms.

Output: An abstraction that depicts all the output mechanisms.

5.3. Refining the Architecture into Top-Level Components

The architecture of the system starts to shape up as we divide the software architecture into its component parts. The point cloud boundary point segmentation tool's top level parts are shown in the diagram below:

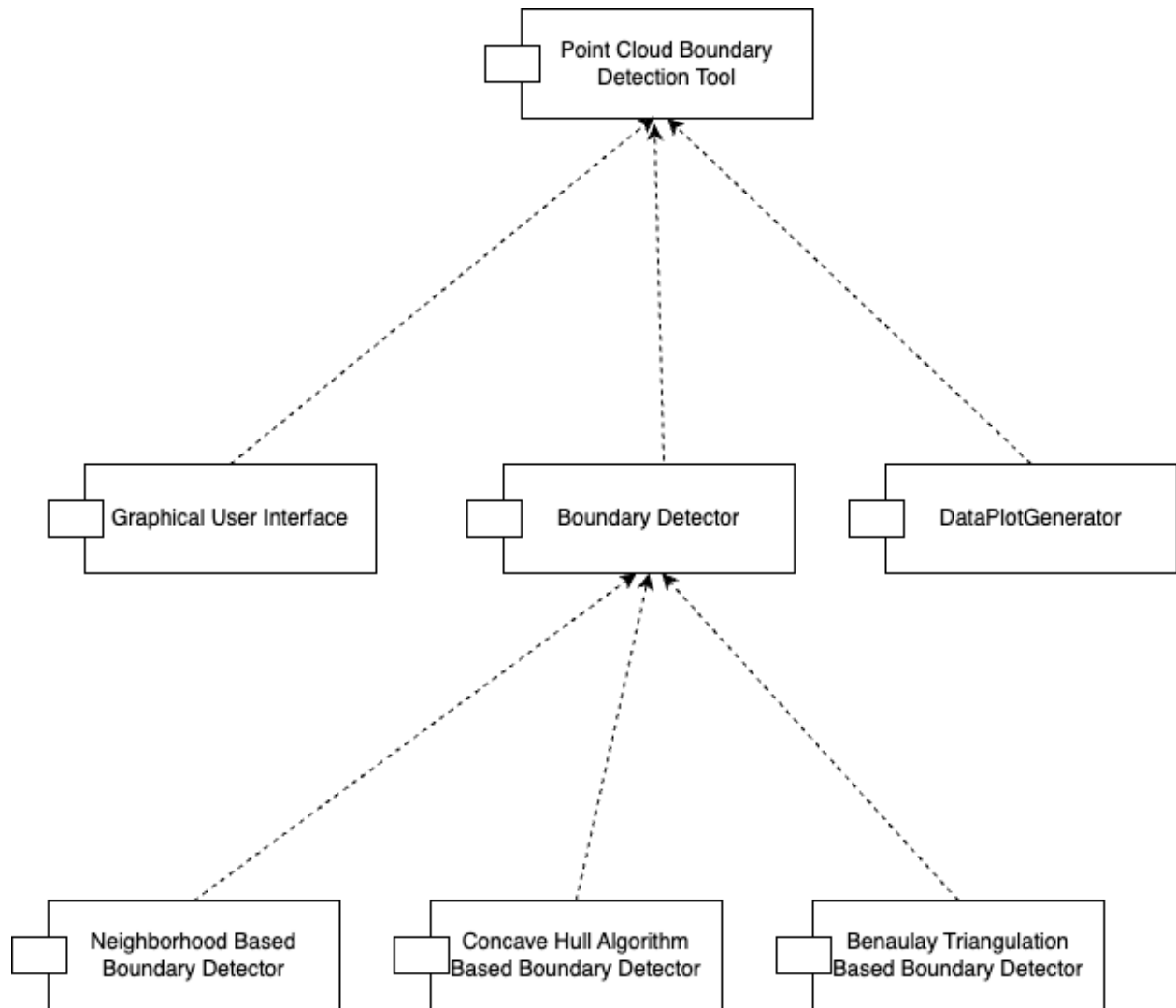


Figure 5: Refining Archetypes to Components

5.4. Mapping Requirements to Software Architecture

According to the requirements, users can input point cloud data and see the boundary highlighted data in graphical user interface or download it as a file. The **User Interface** component will be responsible for taking input and generating output. The **Boundary Detector** component will act as a generalization to apply the algorithms needed to detect boundary points and highlight them to generate output data. The **Data Plot Generator** component will generate

plotting using different libraries according to the user demands. It will involve coloring the point cloud nodes and highlighting the nodes to let the user visualize according the the type of the data.

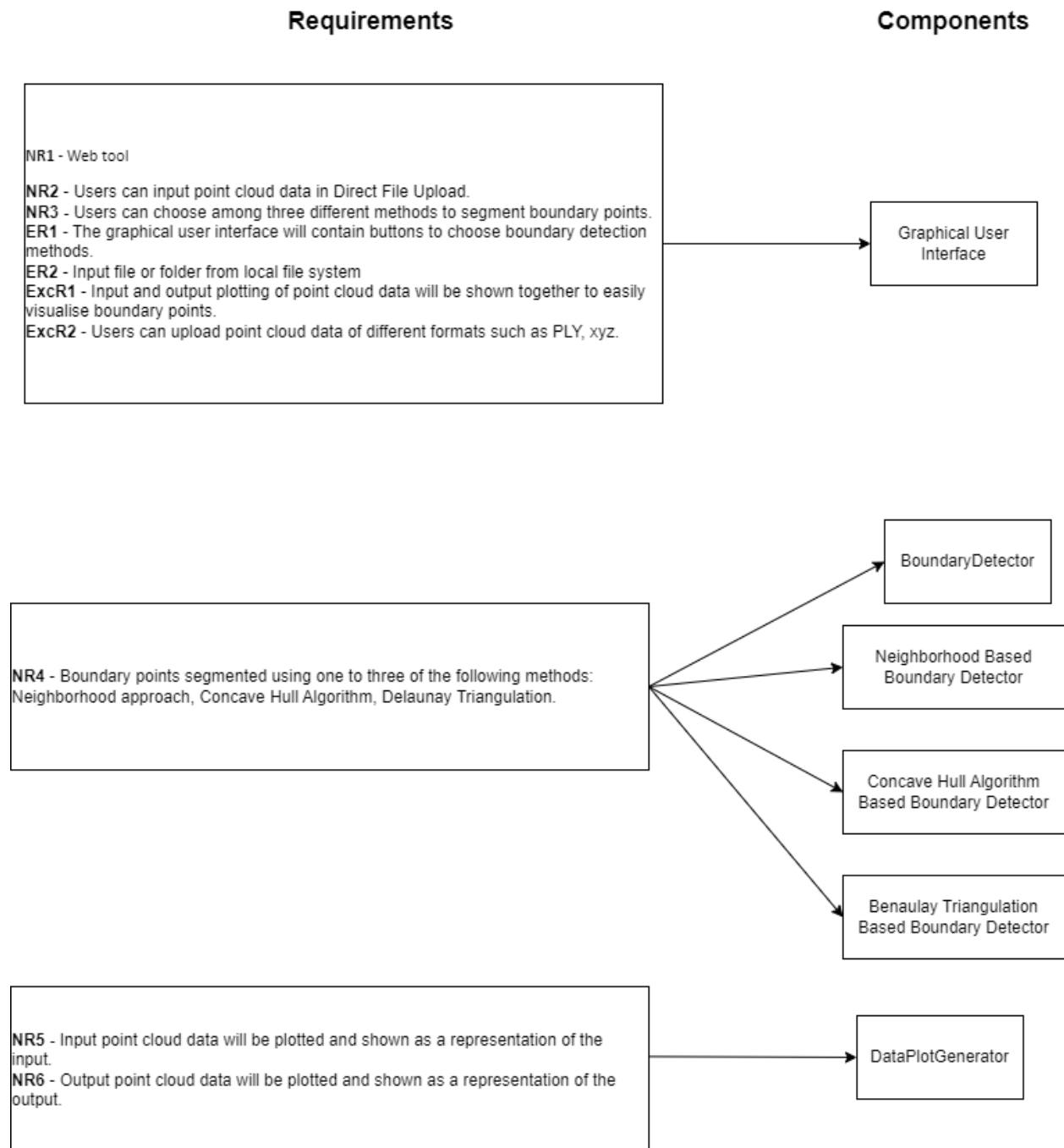


Figure 6: Mapping Requirements to Software Architecture

6. Methodology

This chapter explains the internal mechanisms of the automated boundary point segmentation tool for LiDAR point cloud data.

6.1. Workflow of BorderBlaze

The workflow of the whole project is shown below:

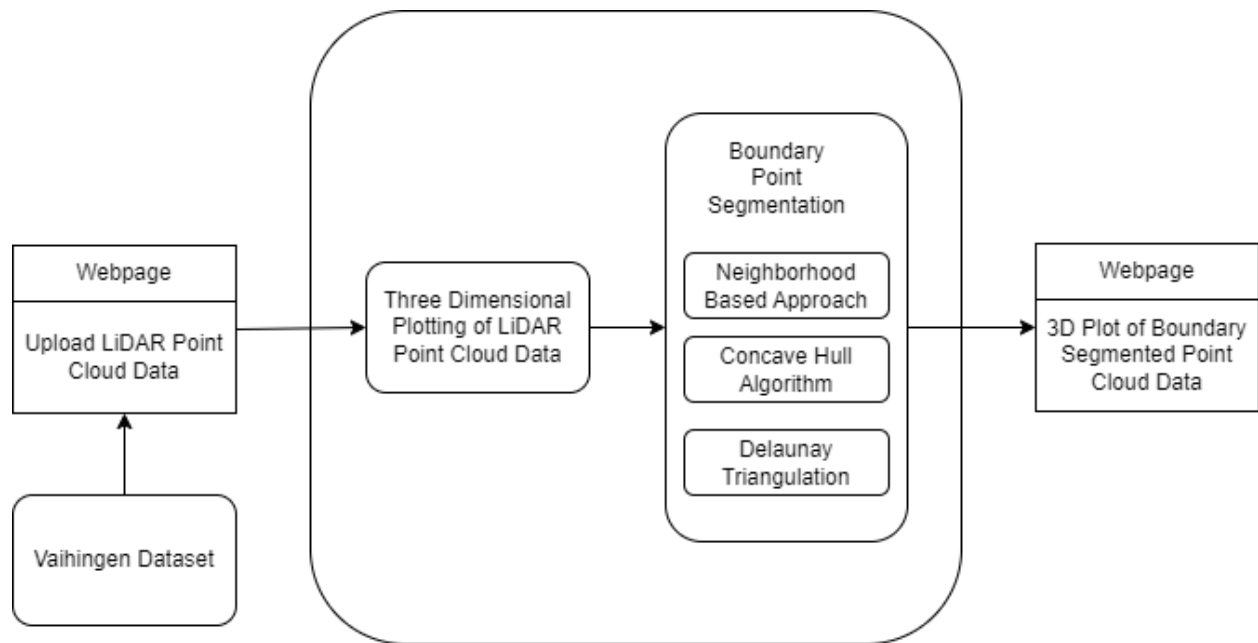


Figure 7: Workflow of BorderBlaze

6.2. Three-Dimensional Plotting of LiDAR Point Cloud Data

We incorporated a robust approach to visualize point cloud data by leveraging the capabilities of Python's open3d library. The necessity for a clear and insightful representation of the point cloud data prompted the adoption of open3d, a powerful tool tailored for such visualization tasks. The process initiated with the loading of the point cloud data, followed by meticulous configuration

of visualization parameters to ensure optimal representation. Subsequently, we harnessed the functionality provided by the open3d library, employing its diverse set of functions to construct a compelling 3D plot that effectively conveyed the spatial characteristics of the data.

The implementation involved a structured sequence of steps, encompassing data preprocessing, library integration, and visualization output. Notably, we delved into the open3d library's features, utilizing specific algorithms and techniques to enhance the visual output and extract valuable insights from the point cloud. Throughout the implementation, we encountered challenges, including data compatibility issues and fine-tuning for optimal visualization. These challenges were systematically addressed through careful debugging and parameter adjustment, ensuring the successful generation of a meaningful 3D plot.

6.3. Boundary Point Segmentation

6.3.1. Neighborhood Based Approach

The segmentation of boundary points in aerial LiDAR point cloud data was executed through a comprehensive Python implementation using the FastAPI framework, NumPy, Matplotlib, and Open3D, complemented by a custom module for generating an empty plot. The process began by leveraging the K-Nearest Neighbors (KNN) algorithm within the K-Nearest Neighbor function to calculate average distances to the k-nearest neighbors for each point, thereby identifying boundary points based on a predefined threshold.

Algorithm The k -nearest neighbors classification algorithm

Input: D : a set of training samples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ k : the number of nearest neighbors $d(\mathbf{x}, \mathbf{y})$: a distance metric \mathbf{x} : a test sample

- 1: **for each** training sample $(\mathbf{x}_i, y_i) \in D$ **do**
 - 2: Compute $d(\mathbf{x}, \mathbf{x}_i)$, the distance between \mathbf{x} and \mathbf{x}_i
 - 3: Let $N \subseteq D$ be the set of training samples with the k smallest distances $d(\mathbf{x}, \mathbf{x}_i)$
 - 4: **return** the majority label of the samples in N
-

Figure 8: KNN Pseudocode

The FastAPI framework facilitated the creation of an interactive web application, and the resulting boundary and non-boundary points were visually represented using the Open3D library, with boundary points displayed in red and non-boundary points in green. A color-coding scheme enhanced the interpretability of the visualization, further aiding in the identification of spatial patterns. The visualization, embedded into an empty plot using a custom function, was made accessible through a dedicated FastAPI endpoint (/method/1/) as a streaming response in PNG format. This methodology showcased a seamless integration of various Python libraries and modules, providing an effective and visually insightful approach to segmenting boundary points in LiDAR point cloud data for enhanced spatial analysis.

6.3.2. Concave Hull Algorithm Approach

The segmentation of a concave hull from a 2D LiDAR point cloud dataset was executed through a Python implementation utilizing the FastAPI framework, NumPy, Matplotlib, and Open3D, along with a custom concave hull module [3]. The process began with the setup of a FastAPI web application, serving as the platform for the visualization of the segmented concave hull. The LiDAR point cloud data, retrieved from a specified file, underwent processing using NumPy, and a custom concave hull algorithm was applied to compute the non-convex shape enclosing the

points. Boundary points of the concave hull were identified, forming distinct red and green point clouds representing the hull and non-hull regions, respectively.

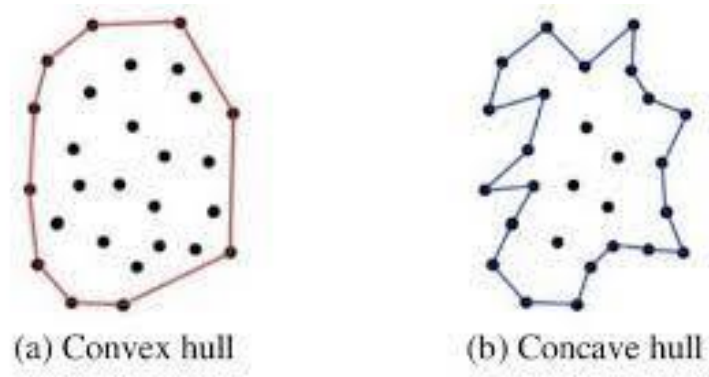


Figure 9: Convex Hull vs Concave Hull Algorithm

Algorithm: Modified Concave Hull Algorithm Pseudocode with Edge Thresholding

```
# Function to Compute Concave Hull with Edge Thresholding
function concave_hull_with_threshold(points, edge_threshold):
    sorted_points = sort_points_by_x(points) # Sort points by x-coordinate
    upper_hull = find_hull(sorted_points, edge_threshold) # Find upper hull with edge
thresholding
    lower_hull = find_hull(reverse(sorted_points), edge_threshold) # Find lower hull with edge
thresholding
    concave_hull_points = concatenate(upper_hull, lower_hull[2 to end-1]) # Concatenate upper
and lower hulls
    return concave_hull_points

# Function to Find Convex Hull with Edge Thresholding
function find_hull(points, edge_threshold):
    hull = [] # Initialize hull list
    for each point in points:
        while len(hull) >= 2 and is_clockwise_turn(hull[-2], hull[-1], point):
            hull.pop() # Remove the last point from hull
        hull.append(point) # Add the current point to hull

# Apply edge thresholding
if len(hull) > 2 and calculate_hull_edge_length(hull) > edge_threshold:
    return hull
else:
    return [] # Discard the hull if its edge length is below the threshold
```

Figure 10: Modified Concave Hull Algorithm

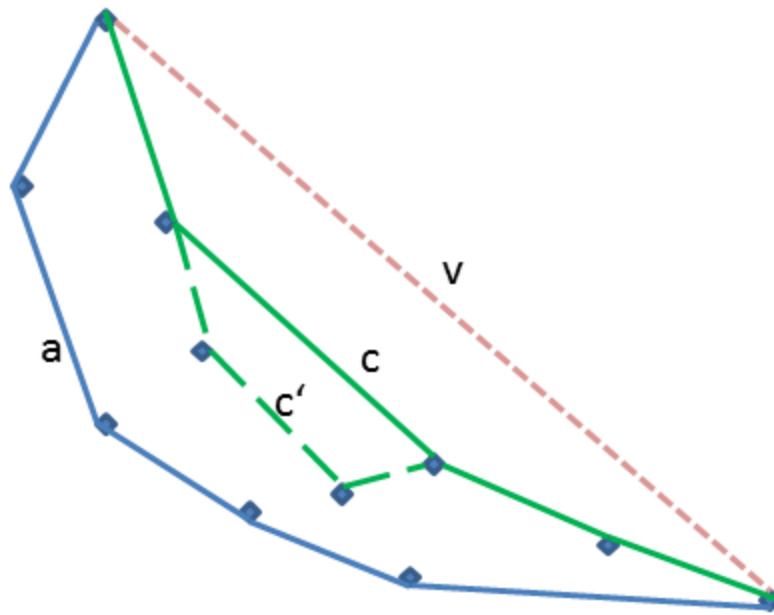


Figure 11: Thresholding based on edge length

Open3D facilitated the creation of 3D point cloud visualizations, showcasing the concave hull's geometry. Additionally, a 2D plot generated with Matplotlib illustrated the concave hull, with blue points representing the original LiDAR data and a red line connecting the boundary points. The resulting visualizations were exposed through a dedicated FastAPI endpoint (`/method/2/`), providing an interactive streaming response for a comprehensive exploration of the segmented concave hull. This methodology demonstrated a holistic approach to concave hull segmentation, emphasizing the integration of diverse Python libraries and modules within a web-based environment for effective spatial analysis and visualization.

6.3.3. Delaunay Triangulation Approach

In the methodology of this project, Delaunay Triangulation [2] has been employed as a foundational geometric algorithm. This technique systematically partitions a given set of points

into non-overlapping triangles, ensuring that no point resides within the circumcircle of any triangle. By maximizing the minimum angle of these triangles, Delaunay Triangulation is known for creating well-conditioned and uniformly shaped triangles. Its application extends across various domains, including computer graphics, terrain modeling, and spatial analysis. In the context of this project, the algorithm serves as a fundamental tool for spatial analysis, contributing to a comprehensive understanding of the geometric relationships and structural features present in the aerial LiDAR point cloud data under investigation.

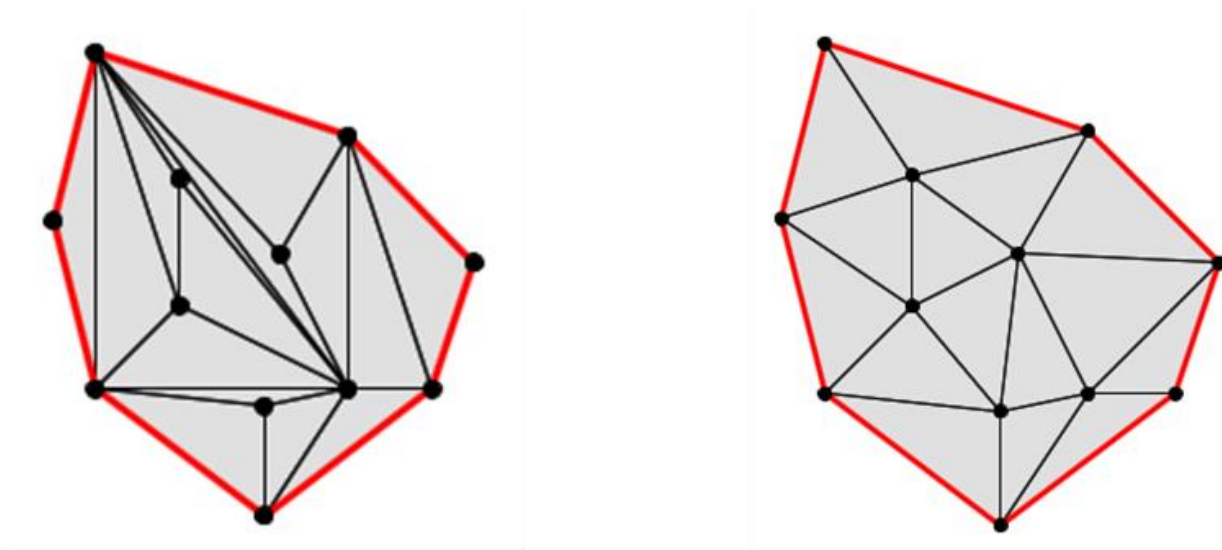


Figure 12: Delaunay Triangulation

Algorithm: DELAUNAY(P, B)

Input: P , a set of points in R^2 ,
 B , a set of Delaunay edges of P which is the border of a region in R^2 containing P .

Output: The set of Delaunay triangles of P which are contained within B .

Method:

1. If all the points in P are on the border B , return END_GAME(B).
2. Find the point q that is the median along the x axis of all internal points (points in P and not on the border). Let \mathcal{L} be the line $x = q_x$.
3. Let $P' = \{(p_y - q_y, ||p - q||^2) \mid (p_x, p_y) \in P\}$. These points are derived from projecting the points P onto a 3D paraboloid centered at q , and then projecting them onto the vertical plane through the line \mathcal{L} .
4. Let $\mathcal{H} = \text{LOWER_CONVEX_HULL}(P')$. \mathcal{H} is a path of Delaunay edges of the set P . Let $P_{\mathcal{H}}$ be the set of points the path \mathcal{H} consists of, and $\bar{\mathcal{H}}$ is the path \mathcal{H} traversed in the opposite direction.
5. Create two subproblems:
 - $B^L = \text{BORDER_MERGE}(B, \mathcal{H})$
 $B^R = \text{BORDER_MERGE}(B, \bar{\mathcal{H}})$
 - $P^L = \{p \in P \mid p \text{ is left of } \mathcal{L}\} \cup \{p' \in P_{\mathcal{H}} \mid p' \text{ contributed to } B^L\}$
 $P^R = \{p \in P \mid p \text{ is right of } \mathcal{L}\} \cup \{p' \in P_{\mathcal{H}} \mid p' \text{ contributed to } B^R\}$
6. Return $\text{DELAUNAY}(P^L, B^L) \cup \text{DELAUNAY}(P^R, B^R)$

Figure 13: Delaunay Triangulation Pseudocode

The delineation of boundary points within an aerial LiDAR point cloud dataset was accomplished through the implementation of a Delaunay triangulation approach encapsulated in the "delaunaytriangulation.py" file. Utilizing the FastAPI framework and key libraries such as NumPy and Open3D, the algorithm processed the LiDAR point cloud data obtained from a designated file, efficiently generating a Delaunay triangulation. Subsequently, the algorithm identified the edges of the triangulation, discerning them as boundary points indicative of structural features within the point cloud. The segmented boundary points were then visualized using Open3D, presenting a clear representation of the delineated boundaries. This methodology highlighted the effective use of computational geometry techniques to discern structural features in aerial

LiDAR data, contributing to a comprehensive understanding of the spatial characteristics of the surveyed area.

6.4. Vaihingen Dataset

The "Vaihingen dataset" refers to a widely used remote sensing dataset that encompasses high-resolution aerial imagery of the Vaihingen region in Germany. From ISPRS benchmark we selected a site from Vaihingen (VH) area with a point density of 2.5 to 3.9 points/m². The dataset contains buildings from residential areas. Using the methods proposed by Dey et al. [1] the individual building area were extracted in existing resources and then we detect the boundary points using our selected methods.

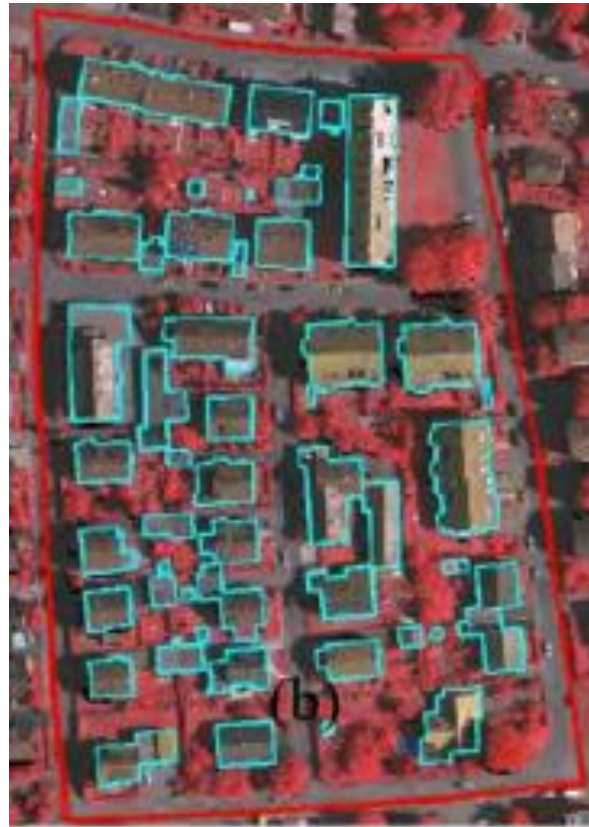


Figure 14: Individual Building Extraction from Vaihingen Dataset

7. User Interface Design

User interface design creates an effective communication medium between a human and a computer. Following a set of interface design principles, design identifies interface objects and actions and then creates a screen layout that forms the basis for a user interface prototype.

Interface Analysis

In the case of user interface design, understanding the problem means understanding

1. The people (end users) who will interact with the system through the interface,
2. The tasks that end users must perform to do their work,
3. The content that is presented as part of the interface, and
4. The environment in which these tasks will be conducted

In the sections that follow, we examine each of these elements of interface analysis with the intent of establishing a solid foundation for the design tasks that follow.

7.1 User Analysis

A tool capable of segmenting boundary points in aerial LiDAR point cloud data would be particularly valuable to users in fields that heavily rely on geospatial information, such as:

Geospatial Analysts and Surveyors:

Professionals involved in geospatial analysis and surveying tasks would benefit from accurate boundary segmentation, aiding in the identification of distinct features and structures within the surveyed area.

Urban Planners and Architects:

Urban planners and architects could leverage such a tool to precisely delineate boundaries, assisting in city planning, infrastructure development, and architectural design.

Environmental Scientists:

Researchers and scientists studying environmental changes and landscapes could utilize the tool to analyze and monitor terrain features, aiding in ecological assessments and conservation efforts.

Civil Engineers:

Civil engineers involved in infrastructure projects, such as road construction or land development, would find value in accurately segmented boundary points for efficient planning and design.

Remote Sensing Researchers:

Researchers focusing on remote sensing applications would use the tool to extract meaningful information from LiDAR point cloud data for various scientific investigations.

7.2. Task Analysis

7.2.1. User Task: Upload aerial LiDAR point cloud data

- Upload file containing aerial LiDAR point cloud data.

7.2.2. User Task: Submit Neighborhood Based Approach Method

- Submit Neighborhood Based Approach Method to view segmented boundary points in the 3D plotting.

7.2.3. User Task: Submit Concave Hull Algorithm Method

- Submit Concave Hull Algorithm Method to view segmented boundary points in the 3D plotting.

7.2.4. User Task: Submit Delaunay Triangulation Method

- Submit Delaunay Triangulation Method to view segmented boundary points in the 3D plotting.

7.3. User Interface and User Manual

User interface layouts with corresponding tasks, objects and actions and how to interact with them are illustrated below:

7.3.1. BorderBlaze Home Page:

User will view the home page after entering the webpage. This screen accommodates **task 1**.

Task 1: Upload aerial LiDAR point cloud data.

BORDER BLAZE

A Boundary Detector Tool for Point Cloud Data

Choose file build10.pts

Upload

Choose an option:

Method 1: Neighborhood Based Approach

Submit

Figure 15: BorderBlaze Home Page

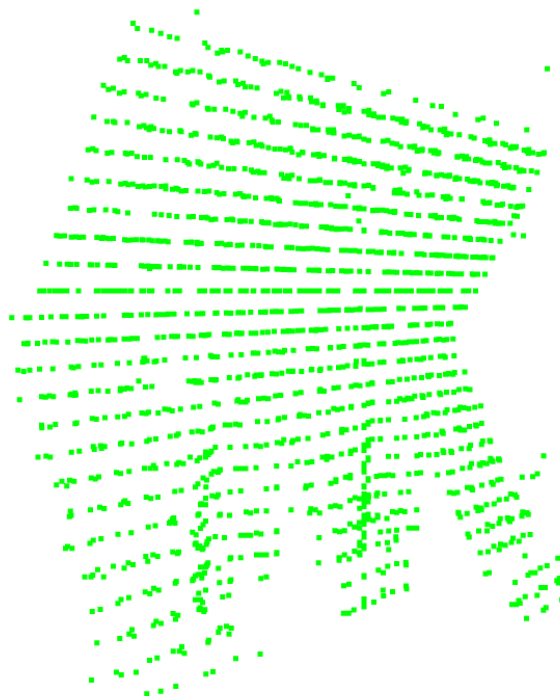


Figure 16: View Input 3D Plotting

7.3.2. View Neighborhood Based Approach Result

After the uploading of aerial LiDAR point cloud file, choose Neighborhood Approach in the dropdown menu and submit to view the results of segmented boundary points using the chosen method. The screen covers **task 2**.

Task 2: Submit Neighborhood Based Approach Method

The screenshot displays the BORDER BLAZE web application interface. At the top right, there is a 'Home' link. The main heading is 'BORDER BLAZE', followed by the subtitle 'A Boundary Detector Tool for Point Cloud Data'. Below this, there is a file upload section with a 'Choose file' button, the filename 'build10.pts', and an 'Upload' button. Underneath the upload section, there is a label 'Choose an option:' followed by a dropdown menu. The dropdown menu is open, showing three options: 'Method 1: Neighborhood Based Approach' (which is selected and highlighted in blue), 'Method 2: Concave Hull Algorithm', and 'Method 3: Delaunay Triangulation'.

Figure 17: Submit Neighborhood Based Approach

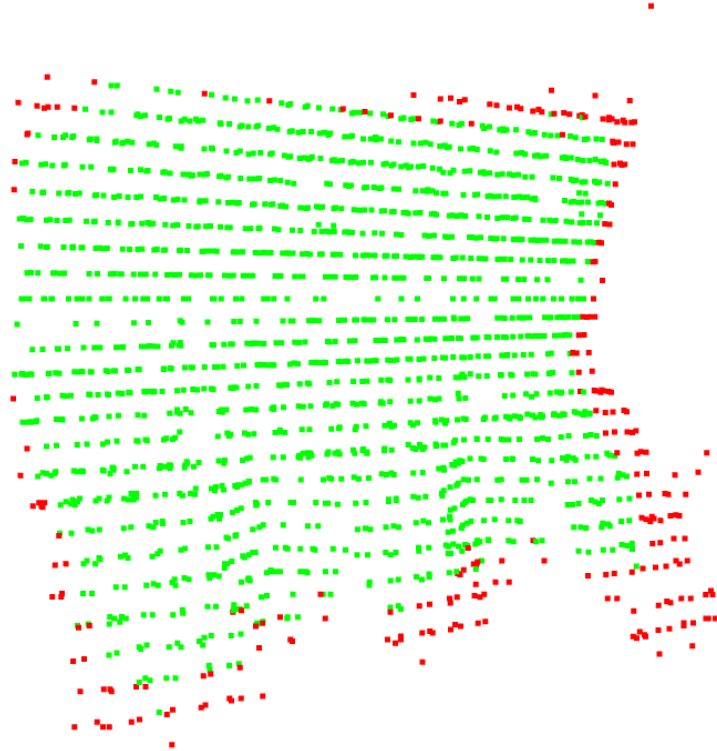


Figure 18: Neighborhood Approach Results

7.3.3. View Concave Hull Algorithm Result

After the uploading of aerial LiDAR point cloud file, choose Concave Hull Algorithm in dropdown menu and submit to view the results of segmented boundary points using the chosen method.

This screen covers **task 3**.

Task 3: Submit Concave Hull Algorithm Method

BORDER BLAZE

A Boundary Detector Tool for Point Cloud Data

Choose file build10.pts Upload

Choose an option:

- Method 1: Neighborhood Based Approach
- ✓ Method 2: Concave Hull Algorithm
- Method 3: Delaunay Triangulation

Submit

Figure 19: Submit Concave Hull Algorithm

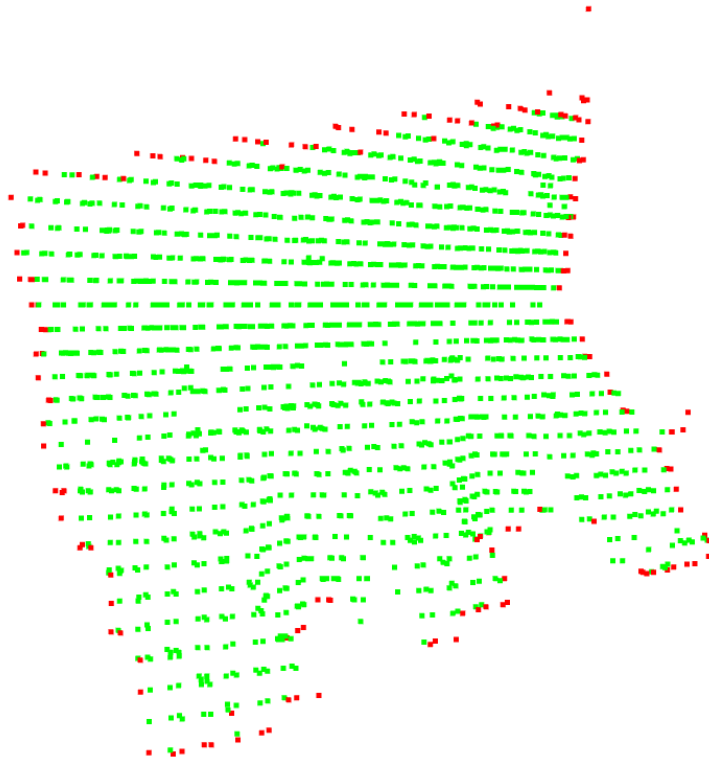


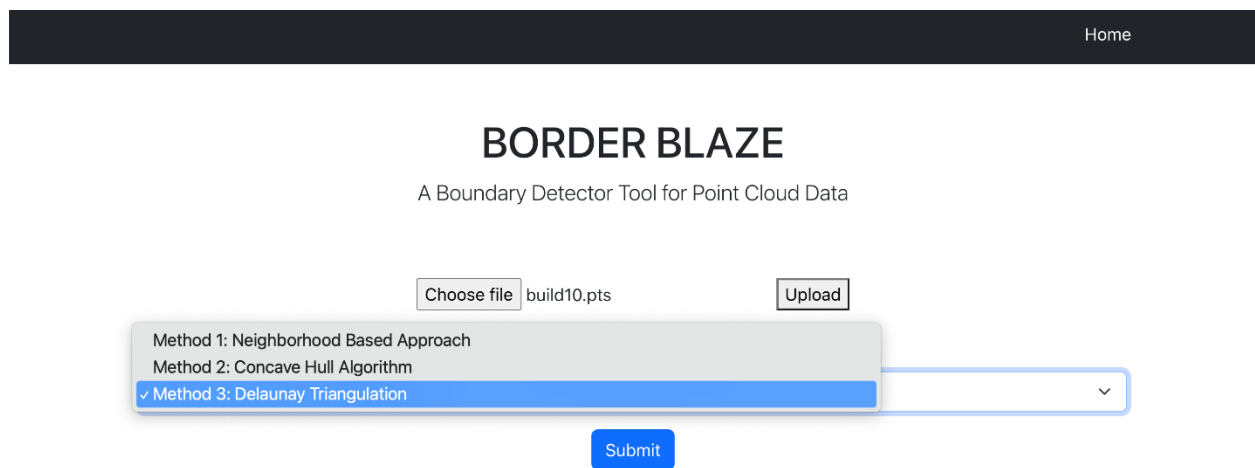
Figure 20: Concave Hull Algorithm Results

7.3.4. View Delaunay Triangulation Result

After the uploading of aerial LiDAR point cloud file, choose Delaunay Triangulation in dropdown menu and submit to view the results of segmented boundary points using the chosen method.

This screen covers **task 4**.

Task 3: Submit Delaunay Triangulation Method



The screenshot shows the BORDER BLAZE web application interface. At the top right, there is a "Home" link. The main heading is "BORDER BLAZE" with the subtitle "A Boundary Detector Tool for Point Cloud Data". Below this, there is a file upload section with a "Choose file" button, the filename "build10.pts", and an "Upload" button. A dropdown menu is open, showing three options: "Method 1: Neighborhood Based Approach", "Method 2: Concave Hull Algorithm", and "✓ Method 3: Delaunay Triangulation". The third option is selected and highlighted in blue. Below the dropdown menu is a blue "Submit" button.

Figure 21: Submit Delaunay Triangulation Method

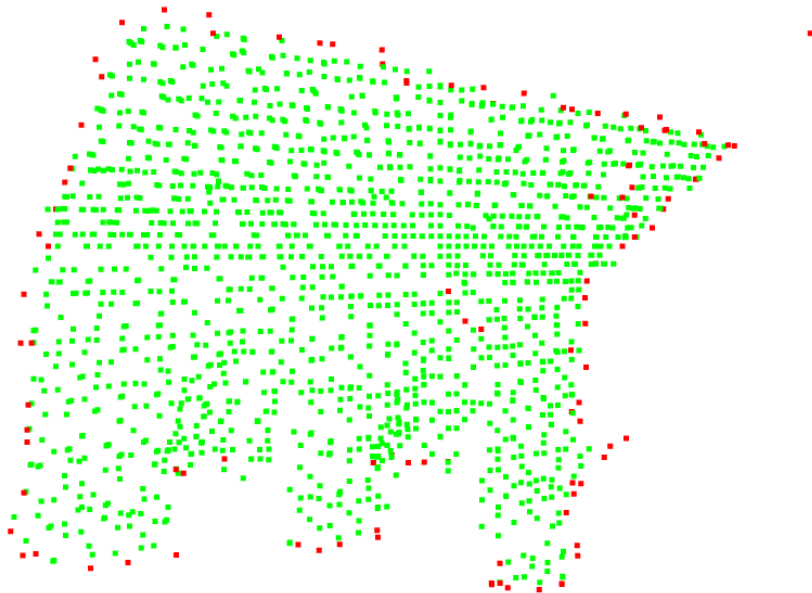


Figure 22: Delaunay Triangulation Method Results

8. Testing

In this chapter, a high-level description of testing goals and a summary of items and features to be tested are presented.

8.1. High-Level Description of Testing Goals

The Point Cloud Boundary Point Segmentation Tool undergoes high-level testing which is popularly known as Black box testing. Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details, and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. Serving as a bridge between users and the development team of a product, the ultimate goal of software testing is to troubleshoot all the issues and bugs as well as control the quality of a product.

Testing goals for this tool is represented below from a high level-

- To demonstrate that the tool meets its requirements
- To ensure input is parsed properly
- To ensure that it produces appropriate stack frames
- To ensure each line, function calls and return types are handled properly
- To ensure that it trace variable changes properly
- To find any existing bugs
- To ensure that the tool runs smoothly

8.2. Test Cases

The item to be tested is a web application. The project is supposed to exist online and can be downloaded from GitHub. All functionalities of the system will be split into modules and black-box testing will be conducted on each feature. Boundary value checking, robustness testing, and worst-case testing will formulate test cases. Input specifications will be created in compliance with the software requirement specification document.

8.3 Item Pass/Fail Criteria

Test ID	Test Cases	Input Test Data	Steps to be executed	Expected Result
T1	Test if a valid Input Works	Input a valid point cloud data format such as xyz.	i) Plot input point cloud data ii) Show options for choosing a method of boundary point segmentation.	Boundary highlighted output data will be plotted and shown.
T2	Test if an invalid input works	Input a PNG image file	i) Alert Message display	Unsupported file format alert message
T3	Test if neighborhood based boundary point segmentation works	i) Input a valid point cloud data ii) Check neighborhood	i) Apply neighborhood based method algorithm	Display boundary highlighted point cloud data

		based boundary point segmentation option.	ii) Highlight boundary points.	plot using neighborhood approach.
T4	Test if concave hull algorithm method works.	i) Input a valid image ii) Check concave hull algorithm method.	i) Apply concave hull algorithm. ii) Highlight boundary points.	Display boundary highlighted point cloud data plot using concave hull algorithm.
T5	Test if Delaunay triangulation method works.	i) Input a valid image ii) Check Delaunay triangulation method.	i) Apply Delaunay triangulation method. ii) Highlight boundary points.	Display boundary highlighted point cloud data plot using Delaunay triangulation method.

9. Conclusion

In this report, I've provided a comprehensive overview of the essential technical aspects related to the development of the "BorderBlaze- Aerial LiDAR Point Cloud Boundary Point Segmentation Tool." The document covers various aspects, including the project's scope, assumptions, Quality Function Deployment (QFD), and usage scenarios. Furthermore, I've employed scenario-based modeling techniques, such as the creation of Use-Case diagrams, Activity diagrams, Class-based diagrams, Methodology and User Interface.

The report also delves into the architectural design and structure of the tool, shedding light on its fundamental components. Additionally, I have performed unit testing and the rest cases are provided in the report. To enhance comprehension, I've incorporated figures and tables throughout the document. Ultimately, my goal with this report is to offer a clear depiction of the entire system's workflow. I trust that this comprehensive document will assist the reader in gaining a thorough understanding of the technical intricacies associated with this tool.

References

- [1] Dey, E. K., Awrangjeb, M., and Stantic, B. (2020). Outlier detection and robust plane fitting for building roof extraction from LiDAR data. *International Journal of Remote Sensing*, 41(16), 6325-6354.
- [2] Yu-ze, N., Ying-lei, C., Lang-bo, Q., Man-yun, H., Zi-hao, Z., & Lei, P. (2016, August). An algorithm of LiDAR building outline extraction by Delaunay triangle. In *Eighth International Conference on Digital Image Processing (ICDIP 2016)* (Vol. 10033, p. 1003302). SPIE.
- [3] Tseng, Y. H., & Hung, H. C. (2016). Extraction of building boundary lines from airborne lidar point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 41, 957-962.
- [4] Li, L., Song, N., Sun, F., Liu, X., Wang, R., Yao, J., & Cao, S. (2022). Point2Roof: End-to-end 3D building roof modeling from airborne LiDAR point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 193, 17-28.
- [5] Awrangjeb, M., Zhang, C., & Fraser, C. S. (2013). Automatic extraction of building roofs using LIDAR data and multispectral imagery. *ISPRS journal of photogrammetry and remote sensing*, 83, 1-18.
- [6] Canaz Sevgen, S., & Karsli, F. (2020). An improved RANSAC algorithm for extracting roof planes from airborne lidar data. *The Photogrammetric Record*, 35(169), 40-57.
- [7] Kwak, E., Al-Durgham, M., & Habib, A. (2012). Automatic 3D building model generation from lidar and image data using sequential minimum bounding rectangle. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39, 285-290.