Automated Asset Management

Author

Ali Sajjad

Muhammad Zubair

Mubashir Hussain

Talha Aslam

Supervisor

Asst Prof Dr Saddaf Rubab

Submitted to the faculty of Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology, in partial fulfillment for the requirements of B.E Degree in Computer Software Engineering.

June 2021

CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled "**Automated Asset Management**" carried out by **_Ali Sajjad, Mubashir Hussain, Muhammad Talha Aslam, Muhammad Zubair_** under the supervision of **_Asst Prof Dr Saddaf Rubab_** for partial fulfillment of Degree of Bachelor of Computer Software Engineering in Military College of Signals, National University of Sciences and Technology, Islamabad during the academic year 2020-2021 is correct and approved.

Approved by

Supervisor

Asst Prof Dr Saddaf Rubab

[illegible]

Date: June 21, 2021

DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification in either at this institute or anywhere else.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signature of Student:

Ali Sajjad

Mubashir Hussain

Muhammad Talha Aslam

Muhammad Zubair

Signature of Supervisor

[illegible]

Asst Prof Dr Saddaf Rubab

Acknowledgements

We would like to thank Allah Almighty for His incessant blessings which have been bestowed upon us. Whatever we have achieved, we owe it to Him, in totality. We are also thankful to our families for their continuous moral support which makes us what we are.

We are extremely grateful to our project supervisor Asst Prof Dr. Sadaf Rubab from MCS who in addition to providing technical help and guidance also provided us moral support and encouraged us throughout the development of the project.

We are highly thankful to our all teachers and staff of MCS who supported and guided us throughout our course work. Their knowledge, guidance and training enabled us to carry out this whole work.

Finally, we are grateful to the faculty of Computer Software Engineering Department of the Military College of Signals, NUST.

In the end we would like to acknowledge the support provided by all our friends, colleagues and a long list of well-wishers whose prayers and faith in us propelled us towards our goal.

DEDICATION

In the name of ALLAH, the most Merciful, the most Beneficent.

Dedicated to our parents and adored siblings whose tremendous support and cooperation led us to this wonderful accomplishment

Abstract

The project aims to develop an automated system for finding the dimensions of the road assets. All this process is being done manually before which required a lot of manpower, time, and cost. But with Automated Asset Management (AAM) all this process is now shifted towards automation and that saves a lot of resources. The Automated Asset Management will help the transportation department in maintaining the road asset`s condition on time. The system will take four input data files with extensions (txt, bin, json, jpg) and it will give you the dimensions of the assets identified in the image.

The system uses LiDAR point cloud data and maps the 3D point data on the 2D image. It will generate an excel sheet in which the information regarding the assets and their dimensions are stored.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

The Automate Asset Management is a system that will be used to measure the dimensions of road assets that are present in the inventory using LIDAR point cloud and Imagery and maintain them in a database.

1.1 Problem Statement

In the most countries where asset management is being done, it is manual using lots of manpower which requires a lot of time and cost fortune.

1.2 Solution

So, for the problem on hand, we propose Automated Asset Management (AAM) in which all the tasks are being automated reducing to manpower required previously to minimum and thus costing a little. The positive side of this system is that error caused by manpower is also reduced to minimum because of accuracy of the system.

1.3 Scope

The Automated Asset Management system calculates the dimensions of the assets. It helps in Road Maintenance Process. The main benefit of this system is that it does all this process automatically rather manually. It saves a lot of time and human resource. When this process is manually done, it takes weeks or sometimes months and a lot of human resource and a lot of cost must be spent on it but with this automation system, a lot of time, cost, and human resource can be saved. It improves the road management and therefore the Transportation.

The assets that are currently present in the inventory, are as follow:

- Streetlight
- Signpost
- Rail guard
- Sidewalk

1.4 Objectives

The main objective of the project is to develop an automated dimensional measurement system which can be integrated into our industrial partner’s existing transportation asset management service which will be used to build a comprehensive asset inventory for the US Department of Transportation and other agencies.

CHAPTER 2: Literature Review

2.1 Methodology

Project from lidar to Cam2:

To convert Velodyne (lidar) points into camera coordinates, consider the following transformation:

$$projection_{matrix} = P2 * R0\_rect * Tr\_velo\_to\_cam$$

Multiplication should be performed in homogenous coordinate to simplify the calculation. To convert to pixel coordinate, simply normalize by z-coordinate.

- In 3D, the homogeneous point (x,y,z,w) corresponds to the Cartesian point (x/w,y/w,z/w)
- There are an infinite number of homogeneous ways to represent each Cartesian point. E.g., (1,1,1,1), (2,2,2,2) and (3,3,3,3) all correspond to the Cartesian point (1,1,1)
- For finite points, w is not 0.

To map points to pixel, this is a projective transformation from lidar to image plane.

- Compute projection matrix
- Project points to image plane
- Remove points that lie outside of image boundaries.

Project from Cam2 to Lidar:

To project a 2d pixel coordinate converted from lidar point in previous section back to its original value:

- Compute inverse of the projection matrix
- Convert 2d point to its homogenous representation.
- Multiply this homogenous point with the inverse of the projection matrix.

$$projection\_matrix = P0 * R0\_rect * Tr\_velo\_to\_cam$$
$$projection\_matrix^{-1} = P0^{-1} * R0\_rect^{-1} * Tr\_velo\_to\_cam^{-1}$$

Finding depth of pixel points:

To project the bounding box points(pixel) to 3d lidar points, the depth of the point is required in cam2/image coordinate system. The depth of the point can be found by comparing the 2d point with the projected lidar points in image coordinates present inside bounding box.

Dimensional Measurements:

The length of the sides of polygon are usually of different sizes.
How to approximate the length and width/height of a polygon? [Problem]
Maybe, approximate a bounding box that can fit a polygon and measure its dimensions.

CHAPTER 3: External Interface Requirements

3.1 User Interfaces

Automated Asset Management (AAM) will be executed from a command line. It will have different command line arguments to control its behavior which will be described in the user manual.

3.2 Hardware Interfaces

- Core i5 or greater
- RAM 4GB Minimum (8 GB recommended)
- VRAM 2GB Minimum (4 GB recommended)

3.3 Software Interfaces

Python v3

For the working of this software, python3 is needed to be installed on the system.

Libraries

Moreover, there are several libraries required to be installed on the system prior than its working:

- NumPy
- SciPy
- openpyxl

Additional libraries will be added later if required.

Furthermore, the additional libraries for data visualization that will be helpful for testing are:

- matplotlib
- mayavi
- OpenCV-python
- PyQt5

OS permission

Regarding OS permission control, access must be granted to read and write the files so that software can work uninterruptedly.

CHAPTER 4: System Features

This segment of this document will be explaining features of the AAM software, their stimulus and responses, sequence of actives (if any) and functional requirements related to these features.

4.1 Validating Input

Description and Priority

The system will validate the input dataset according to the provided standard format. A single instance in dataset consists of four files given below with their corresponding extension:

- Lidar data (.bin)
- Image (.png)
- Annotated assets data (.json)
- Calibration (.txt)

An instance of dataset will have the same name plus the appropriate extension. For example, 000040.bin -- 000040.png -- 000040.json -- 000040.txt
An additional ground-truth file (.csv) can be provided as an input for evaluation process.

Stimulus/Response Sequences

After feeding the system with input dataset, it will validate the first instance of the data set. If the data instance is validated successfully, the system will move to the next step of dimensional measurements. If the data instance is not validated successfully, then the system will store an appropriate error message and will move to the next data instance in the set if any, or else the system will exit after generating an output file. An additional step of evaluation will be performed before exiting if the ground truth is provided.

Functional Requirements

REQ-1.1: The system should be able to read the dataset from a given directory.
REQ-1.2: The system should be able to validate the instance of dataset before moving to the next step. If that instance contains any error, record an appropriate message for that error in the final output file.

4.2 Dimensional Measurement of the Assets

Description and Priority

After the data instance is validated successfully, it goes through a series of steps to measure the dimensions of the assets present in the inventory. And then records it in the final output file (excel spreadsheet).

Stimulus/Response Sequences

The validated data instance is passed to the dimensional measurement step. If the dimensions are calculated successfully, they are recorded in the output file. If not, then the appropriate error is recorded in the output file. And then the next data instance goes through the same series of steps if any in the dataset. An additional evaluation step might be performed before exiting (same step as described in section 4.1.2).

Functional Requirements

REQ-2.1: The system should be able to measure the 2d-dimensions (length and width) of the assets present in the inventory.

REQ-2.2: The system should be able to record the successful dimensional measurements in the output file.
REQ-2.3: The system should be able to record appropriate error messages for the unsuccessful dimensional measurements in the output file.

4.3 Evaluation

Description and Priority

It is an optional step to evaluate the system accuracy against the ground-truth of dataset provided to the system. It will be controlled via command line argument.

Stimulus/Response Sequences

After the dimensional measurements of all the instances of the dataset, the accuracy of the system can be measured from comparing the ground-truth provided as an input via command line and results of the system. The system will generate an evaluation results file before exiting. It will include the overall combined accuracy of the system and the accuracy of each class of the assets present in the inventory and some additional statistical analysis.

Functional Requirements

REQ-3.1: The system should be able to evaluate itself from the ground-truth of dataset provided via command line.
REQ-3.2: The system should be able to record the evaluation results in a file. It should include the overall accuracy of the system and the accuracy of each class of the assets present in the inventory.

CHAPTER 5: Nonfunctional Requirements

5.1 Performance Requirements

- Software should be optimized enough to use least resources required for computations.
- Software should be time effective so that it speeds up the process.

5.2 Safety Requirements

- Exceptions should be handled well in the software so that unexpected crashes can be avoided.

5.3 Software Quality Attributes

- The final Python script should be properly commented.
- The code should follow the PEP-8 style guide for Python code.

CHAPTER 6: Architecture

6.1 Architecture Design

The architectural design of the Automated Asset Management is Pipe and Filter. The pipe and filter divide the system into the following modules to achieve the complete functionality.

Pipe
The pipes serve as connectors for the stream of data being transformed, each connected to the next component in the pipeline.
Filter
Filters are the independent entities also known as components, which performs transformation on data and process the input they receive.

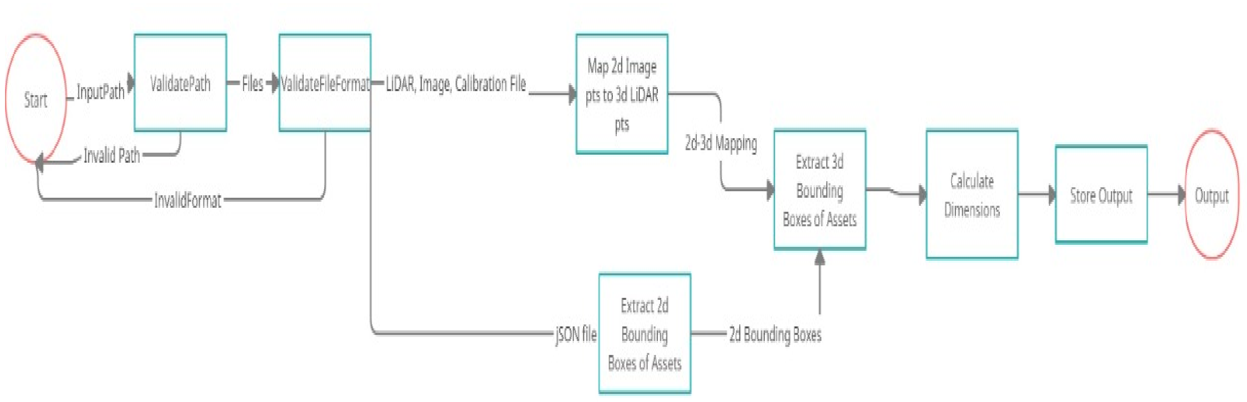


Figure 1System Architecture

6.2 Decomposition Description

The decomposition of the system is explained in the following two ways.

Module Decomposition

NIL

Process Decomposition

The process decomposition is explained through sequence diagram which decomposes the system into well-defined and cohesive processes. The sequence diagram shows the sequence of processes that undertakes to give the desired output.

6.3 Sequence Diagram:

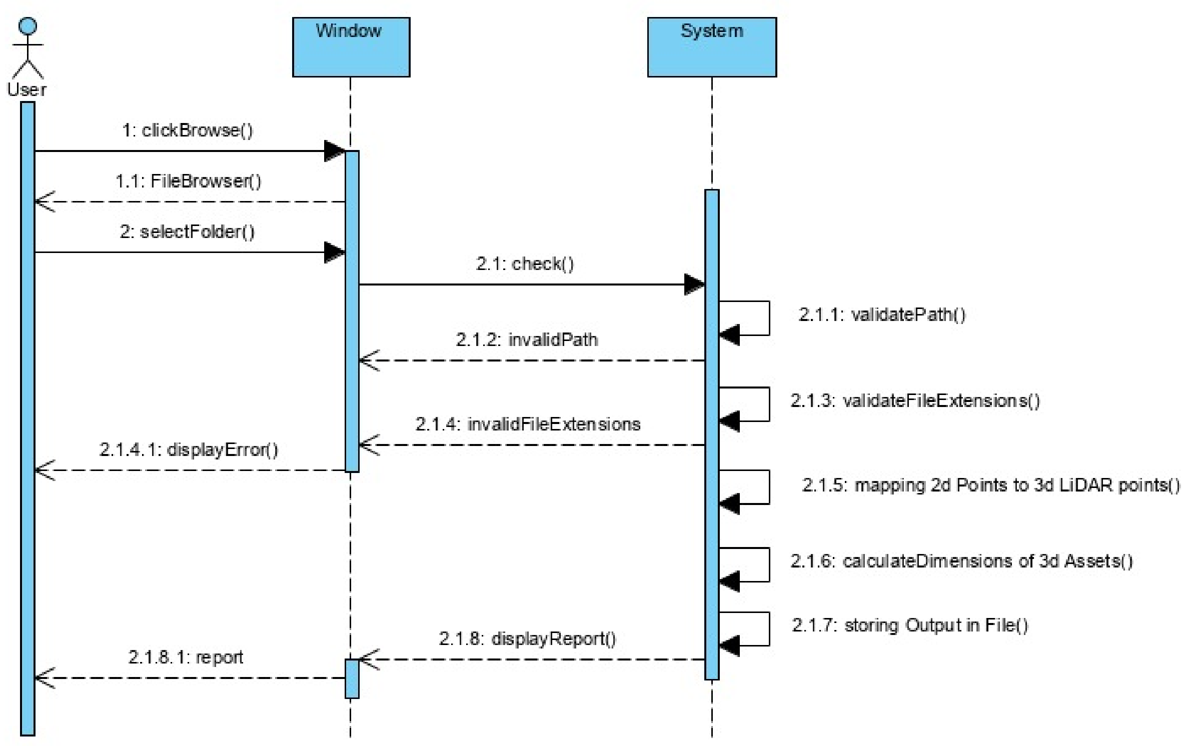


Figure 2 Sequence Diagram

The sequence diagram shows the sequence of events and interactions arranged in time sequence from user's perspective. The user will give input to the system and then the system will validate the input if the input is incorrect then it will display an error to the user otherwise if input is correct then system do the mapping of 2d image to 3d Lidar points and after that calculate the dimensions of the assets present in the image and record these dimensions onto an excel file and display that file to the user at the termination of the program.

6.4 Design Rationale

The architecture chosen for the Automated Asset Management is Pipe and Filter.
It has independent entities called filters (components) which perform transformations on data and process the input they receive, and pipes that serve as connectors for the stream of data being transformed, each connected to the next component in the pipeline. As the process of calculating

dimensions is quite large and requires a lot of data processing and transformation, so it is broken down to multiple steps. Each filter is responsible for one of the steps and this architecture supports parallelism which can increase the speed of the system. We are giving a directory/path to the program as input and that directory contains many files. So, if path or format of the file isn`t correct than the program won`t stop there, it shows error to that input and moves on to the next one. Moreover, each filter is responsible for applying a function to the given data and some filters run parallel to each other to give the desired output.

CHAPTER 7: Data Design

7.1 Data Description

The training dataset for the automated asset management system comprised of four files for one component/image.

File Name	File Extension	File Description
JSON	.json	It contains bounding boxes of assets which includes their vertices and names of assets.
TEXT	.txt	It contains camera calibrations matrix for images with LIDAR camera as reference.
BIN	.bin	It contains the 3D point cloud but in binary form.
IMAGE	.png	It is a 2D image of roadway.

Table 1Data Description

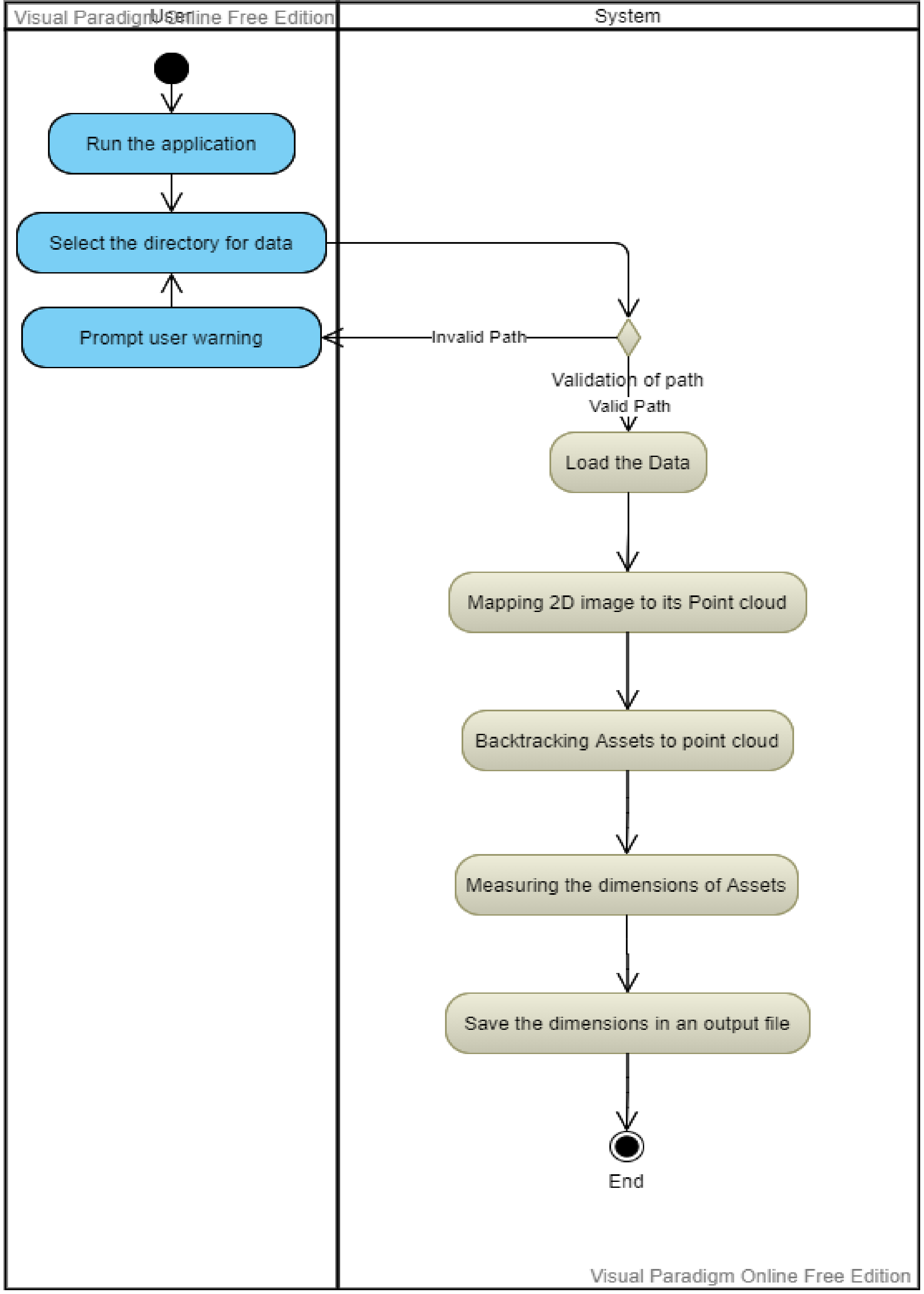
7.2 Data Dictionary

Function Table

Function Name	Parameters	Description
load_lidar	file_path	Loads lidar data from directory.
load_image	file_path	Loads image from directory.
load_calibration	file_path	Loads the camera calibrations from file path.
load_assets	file_path	Loads Bounding boxes from file path in json file.
load_ground_truth	file_path	Loads ground truth values from file path for comparison
load_data_instance	base_file_name, path=DEFAULT_DATA_PATH	
get_projection_matrix	calib_dict	Calculates the projection matrix from calibration dictionary.
project_lidar_to_image	point_cloud, projection_matrix, image_shape	Project the lidar points to image using calculated projection matrix.
get_points_in_polygon	polygon_points, test_points	Gets the points in the polygon by comparing its depth with other points in the polygon.
polygon_area	polygon_points	Finds the area of the polygon using its given vertices.
get_depth	asset_label, asset_points, asset_lidar_points	Finds the depth of a point in 2D image with using mapped lidar points.
project_image_to_lidar	asset_points, inv_projection_matrix	Maps the images to their point cloud.
reduce_dimensions	asset_points_3d	Reduces the dimension from 3D to 2D, mainly used for side walks
minimum_bounding_rectangle	asset_points	Finds the rectangle around given points with minimum area.
main		Controls whole program

Table 2 Data Dictionary

CHAPTER 8: Component Design



User
System
Run the application
Select the directory for data
Prompt user warning
Invalid Path
Validation of path
Valid Path
Load the Data
Mapping 2D image to its Point cloud
Backtracking Assets to point cloud
Measuring the dimensions of Assets
Save the dimensions in an output file
End
Visual Paradigm Online Free Edition
Visual Paradigm Online Free Edition

Figure 3System's Component Diagram

In this section we will provide the detailed description of the functions listed in section 7.2.

Function 1

Name:
load_lidar
Dependencies:
Full path including filename of lidar point cloud binary (.bin) file.
Description:
Loads lidar point cloud data from a binary file into numpyndarray.
Summary:

Returns
----
point_cloud: numpy.ndarray
An [N x 3] point cloud matrix

Raises

FileNotFoundError
If file is not found at specified path
ValueError
If the specified file is not in raw binary format (.bin)

Function 2

Name:
load_image
Dependencies:
Full path including filename of image file.
Description:
Loads image into numpyndarray.
Summary:
Returns
----
image: numpy.ndarray
[height x width x channels] image pixel value matrix

Raises

FileNotFoundError
If file is not found at specified path
UnidentifiedImageError
If the specified image file is not supported by Pillow

Function 3

Name:
load_ calibration
Dependencies:
Full path including filename of calibration file.
Description:
Loads calibration data into a dictionary.
Summary:
Returns
----
calib_dict: {str: numpy.ndarray}
A dictionary of calibration data with name as key and values as numpy array

Raises

FileNotFoundError
If file is not found at specified path
IncorrectFormatError
If the specified calibration file is not in proper Kitti format
ValueError
If the file contains numeric values in incorrect format

Function 4

Name:
load_ assets
Dependencies:
Full path including filename of assets' label file.
Description:

Loads assets' data into a dictionary.
Summary:
Returns
----
asset_dict: dictionary (json)
A dictionary of assets' data (labels, points)
Raises

FileNotFoundError
If file is not found at specified path
JSONDecodeError
If there is any error in decoding json data

Function 5
Name:
load_ground_truth
Dependencies:
Full path including filename of ground truth file.
Description:
Loads ground truth csv file into numpy array.
Summary:
Returns
----
ground_truth: numpy.ndarray
A 2d numpy array of ground truth data
Raises

FileNotFoundError
If file is not found at specified path
ValueError
If the file contains formatting and value errors

Function 6
Name:
load_data_instance
Dependencies:
Base file name of the data instance, like '000040' from '000040.png'.
Path of dataset, it will look for files in the following folders in
the provided path
'path/images' for image (png) files
'path/bin' for lidar (bin) files
'path/calib' for calibration (txt) files
'path/assets' for assets (json) files
Description:
Loads an instance of data from the given path.
Summary:
Returns
----
image, point_cloud, calib_dict, asset_dict: tuple
A tuple consisting of the above data if the instance is loaded
Successfully.
Raises

See load_image, load_lidar, load_calibration and load_assets documentation.

Function 7
Name:
get_projection_matrix
Dependencies:
calib_dict: {str: numpy.ndarray}
A dictionary of calibration data with name as key and values as numpy array

Description:
Get lidar points to camera-2 image points projection matrix and its inverse matrix.
Summary:
Returns
----
projection_matrix: np.ndarray
[3 x 4] Lidar to camera-2 image projection matrix
inv_projection_matrix: np.ndarray
[4 x 4] Inverse matrix of projection_matrix (Homogeneous Coordinate System)

Function 8
Name:
project_lidar_to_image
Dependencies:
point_cloud: np.ndarray
[N x 3] point cloud matrix
projection_matrix: np.ndarray
[3 x 4] Lidar to camera-2 image projection matrix
image_shape: tuple/list
A tuple/list of image's height, width in sequence
Description:
Projects lidar point-cloud to camera-2 image points.
Summary:
Returns
----
points_2d: np.ndarray
[N x 3] lidar points matrix in camera-2 coordinates. Only points in image's fov (field of view) are returned.
indices: np.ndarray
1D array of indices of lidar points in image's fov, so that they can be indexed in the original point_cloud matrix.

Function 9
Name:
get_points_in_polygon
Dependencies:
polygon_points :np.ndarray
[N x 2] asset/polygon bounding box/vertices matrix
test_points :np.ndarray
[N x 3] lidar points matrix already coverted to camera-2 cooridnates
Description:
Finds out which 2d test points are lying inside the boundary of the given polygon..
Summary:
Returns
----
points_in_polygon: np.ndarray
[N x 3] matrix of lidar points (in camera-2 space) lying inside the boundary of the polygon/asset's bounding box.

Function 10
Name:
polygon_area
Dependencies:
polygon_points: np.ndarray
[N x 2] asset/polygon bounding box points/vertices matrix.
Description:
Finds area of the polygon.
Summary:
Returns
----

poly_area: float
    Area of the polygon

Function 11

Name:
get_depth
Dependencies:
asset_label: str
    Name of the asset
asset_points: np.ndarray
    [N x 2] asset's bounding box points matrix
asset_lidar_points: np.ndarray
    [N x 3] matrix of lidar points (in camera-2 space) lying inside the
    boundary of the asset's bounding box.
Description:
Finds the depth (z-coordinate) of the asset's bounding box points from the lidar points in camera-2 image space enclosed in bounding box of the asset.
Summary:
  Returns
  ----
asset_points_depth: np.ndarray
    [N x 3] asset bounding box points matrix with depth as third coordinate.
idx_list: np.ndarray
    Indices of closest points in asset_lidar_points to asset_points

Function 12

Name:
project_image_to_lidar
Dependencies:
asset_points: np.ndarray
    [N x 3] asset coordinates in image space including depth as third coordinate.
inv_projection_matrix: np.ndarray
    [4 x 4] inverse of lidar to camera-2 image projection matrix.
Description:
Project`s camera-2 image/asset points to lidar point-cloud.
Summary:
  Returns
  ----
asset_points_3d: np.ndarray
    [N x 3] asset points in lidar point-cloud space

Function 13

Name:
reduce_dimensions
Dependencies:
asset_points_3d: np.ndarray
    [N x 3] matrix of points in 3d space
Description:
Projects 3d points into 2d space using Singular Value Decomposition and takes centroid of 3d points as origin in 2d space.
Summary:
  Returns
  ----
asset_points_2d: np.ndarray
    [N x 2] matrix of points in 2d space projected from the points in the new 3d space.

Function 14

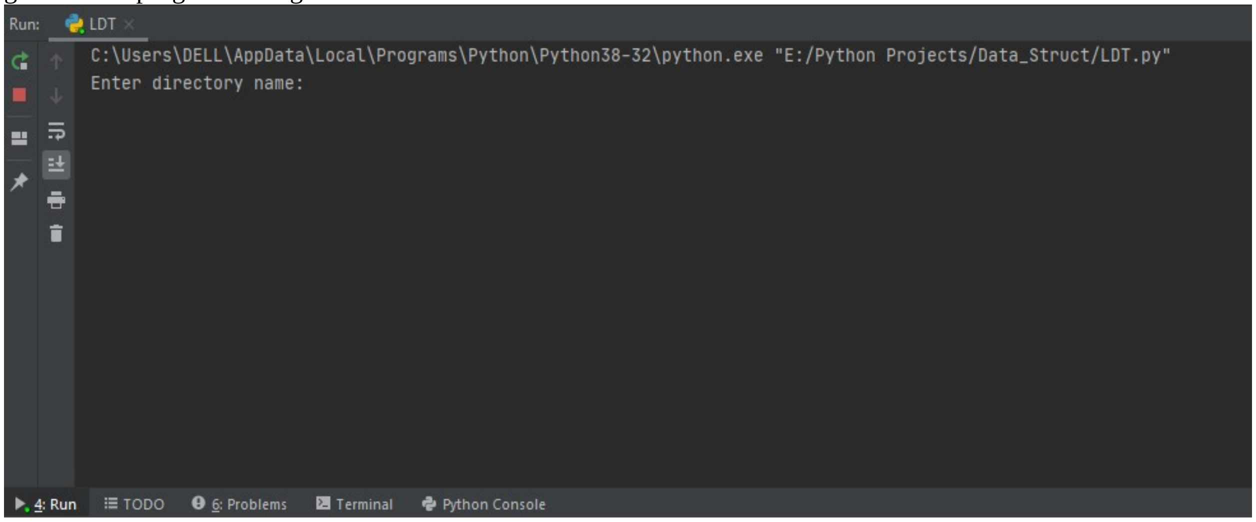
Name:
minimum_bounding_rectangle
Dependencies:
asset_points: np.ndarray

[N x 2] matrix of asset points in 2d space
Description:
Find the smallest bounding rectangle for a set of points.
Summary:
Returns
rect_points: np.ndarray
[4 x 2] matrix of minimum bounding rectangle points

CHAPTER 9: Human Interface Design

9.1 Overview Of User Interface

The User Interface of this program is Command Line Interface (CLI). When user runs the program, the input will be given to the program through command line.



```
C:\Users\DELL\AppData\Local\Programs\Python\Python38-32\python.exe "E:/Python Projects/Data_Struct/LDT.py"
Enter directory name:
```

Figure 4 User Interface

CHAPTER 10: Requirements Matrix

Sr	**Requirement**	**Component**
1	User will enter the input directory and system will validate the path and load the data if corrected.	Function 1,2,3,4,5,6
2	After loading the data, system will project 3D lidar points on image and after doing several processes, find the dimensions of the assets.	Function 7,8,9,10,11,12,13,14
3	After finding the dimensions, system will compare the results with the ground truth and find the absolute error.	Main Function

APPENDIX A

REFERENCES

https://wrf.ecse.rpi.edu/pmwiki/pmwiki.php/Main/HomogeneousCoords
https://www.mrt.kit.edu/z/publ/download/2013/GeigerAl2013IJRR.pdf
https://www.programmersought.com/article/67871412286/
https://medium.com/test-ttile/kitti-3d-object-detection-dataset-d78a762b5a4
https://github.com/yanii/kitti-pcl/blob/master/KITTI_README.TXT
https://medium.com/swlh/camera-lidar-projection-navigating-between-2d-and-3d-911c78167a94
https://stackoverflow.com/questions/45333780/kitti-velodyne-point-to-pixel-coordinate
https://github.com/bostondiditeam/kitti/blob/master/Papers_Summary/Geiger2013IJRR/readme.md
https://docs.enthought.com/mayavi/mayavi/mlab.html
https://gis.stackexchange.com/questions/93848/finding-length-and-width-of-polygon-using-qgis

Plagiarism Report

Automated Asset Management

ORIGINALITY REPORT

18%	11%	0%	15%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	9%
2	wrf.ecse.rpi.edu Internet Source	1%
3	medium.com Internet Source	1%
4	Submitted to University of Pretoria Student Paper	1%
5	Submitted to University of Bahrain Student Paper	1%
6	Submitted to RDI Distance Learning Student Paper	1%
7	Submitted to Colorado Technical University Online Student Paper	1%
8	Submitted to 於2012-06-13提交□ Higher Education Commission Pakistan Student Paper	<1%

9	www.slideshare.net Internet Source	<1%
10	Submitted to CSU, San Jose State University Student Paper	<1%
11	Submitted to University of Maryland, University College Student Paper	<1%
12	Submitted to University of Western Ontario Student Paper	<1%
13	eprints.kfupm.edu.sa Internet Source	<1%
14	id.scribd.com Internet Source	<1%
15	syedhasan010.medium.com Internet Source	<1%
16	Mahesh Shirole, Rajeev Kumar. "UML behavioral model based test case generation", ACM SIGSOFT Software Engineering Notes, 2013 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		