Hackathon Introduction

Welcome to the Hackathon. Please read this file very carefully and make
sure all the examples work for you before the event starts.

Highway Dataset

Besides this software package you should download the dataset. For the examples to work please extract it in this folder under "highway_data". That means you have two folders here "highway" and "highway_data".

Here is a brief explanation of the dataset:

splitted/reduced_*_intensity.ply

The are pointcloud files of different segments of the highway. In total we have 54 segments, each segment is about 100 meter long. The files contain besides the points also intensity information as colors. This is the laser intensity the LIDAR detected when sample the highway at each of the points. It corresponds the reflectivity of that surface at this point and is roughly the brightness to the human eye.

To view these files we recommend using Meshlab as described below. For that to work it is however necessary to created centered ply files.

planar?/sideview_??????_??????.jpg

The vehicle has 3 cameras, each stored in *planar1* to *planar3*. Each camera has a different angle and these are the images taken.

planar?/reference.json (and *.csv)

These file contains the position and poses, as well as gps time of each of the frames. With this information one can project the 3D points of the pointclouds into the cameras. See Example 2 for more information.

The *.csv file contains the same information as a table format.

splitted/reduced*.json (and dataset.json)

These files contain the poses of this segment. They are optional material you might use or not.

The `dataset.json` file contains all these poses in one file.

trajectory.csv

This file contains the positions and timestamps of the LIDAR sensor itself. It has more resolution than the cameras and might be useful to reconstruct a more fine trajectory of the vehicle.

Software Installation

Please make sure you install all the required software for the best experience.

Python

While it is not strictly necessary that you use Python for this task, we have provided a software package to make your life a lot easier in Python. So it makes sense to at least set it up and see how things are done.

Windows

go to https://www.python.org and install Python version `3.8.9`. Version `3.9` doesn't have the `Open3d` package and won't work.

Linux

On Linux Python should already be installed. Just run `python3 -V` on the terminal and check the version. Make sure you have a version `3.6`, `3.7` or `3.8`. Otherwise please install one from sources. Also make sure you have the following packages installed:

```
sudo apt install python3-pip
sudo apt install python3-tk
```

We have tested this setup for both Ubuntu 20.04, 20.10 and Windows 10 with `Python 3.8`.

Python Packages

Make sure to install all the following python packages. They are used for different purposes:

- **numpy:** for vector and matrix computations
- **numba:** for compiling python code to C++ to speedup tight loops
- **OpenCV 2:** for image processing, reading and writing images
- **matplotlib:** for plotting graphs and images
- **open3d:** for displaying and processing pointcloud
- **pyproj:** for converting between pointcloud XYZ and GPS coordinates
- **folium:** for creating interactive maps based on OpenStreemMaps

To install all of these packages please run the following commands:

On Ubuntu:

```
python3 -m pip install --upgrade pip setuptools wheel
python3 -m pip install --upgrade -r requirements.txt
```

On Windows:

```
py -3.8 -m pip install --upgrade pip setuptools wheel
py -3.8 -m pip install --upgrade -r requirements.txt
```

Windows Microsoft Visual C++ Redistributables

For some of the extensions the `Microsoft Visual C++ 2015-2019 Redistributable` needs to be installed. For most installations this is already the case. If your install is however very fresh and you get errors regarding DLL import, you need to install it if necessary. It can be found via Google very easily and afterwards shows up under `Control Panel -> Uninstall Software`.

Meshlab

Meshlab is used to view the pointclouds.

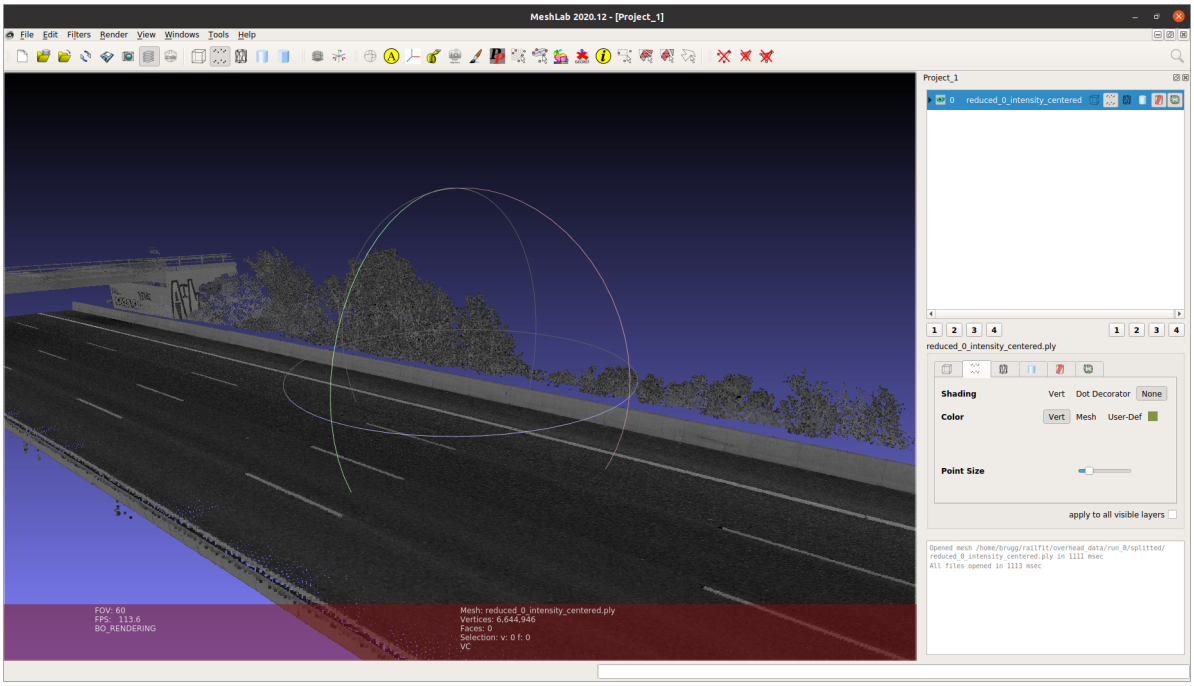
For Windows please download it from here https://www.meshlab.net/

For Linux it is best to install the snap. This is the command for Ubuntu:

```
sudo snap install meshlab
```

You can now open `*.ply` files. If you see artifacts you need to created centered ply files. The reason is that the pointclouds are rendered on the GPU as single precision floating point number. And our coordinates are just to large. See below on how to created these centered ply files.

To see the intensity you need to select **Shading = None** on the right panel. You can import multiple `*.ply` files and this is very useful when debugging your code. You can export points / meshes and see how the fit. Also you can import multiple highway segments.



Create Centered PLY Files

The first thing you should do is to created centered ply files. For that simply run

```
python3 center_ply.py
```

This will run for a few minutes and generate the files `highway_data/splitted/reduced_*_intensity_centered.ply`. These files can then be viewed in Meshlab.

Also this file creates the file `highway_data/splitted/center.txt`. This is the center that was subtracted from each pointcluod. If you work on the original files and want to debug them, you should also subtract this center, so its in the same coordinate system.

```
(env) brugg@ubuntu:~/railfit/dyn_cleareance_gauge/hackathon$ python3 center_ply.py
highway_data/splitted/reduced_0_intensity.ply
highway_data/splitted/reduced_10_intensity.ply
highway_data/splitted/reduced_11_intensity.ply
highway_data/splitted/reduced_12_intensity.ply
highway_data/splitted/reduced_13_intensity.ply
highway_data/splitted/reduced_14_intensity.ply
highway_data/splitted/reduced_15_intensity.ply
highway_data/splitted/reduced_16_intensity.ply
highway_data/splitted/reduced_17_intensity.ply
highway_data/splitted/reduced_18_intensity.ply
highway_data/splitted/reduced_19_intensity.ply
highway_data/splitted/reduced_1_intensity.ply
highway_data/splitted/reduced_20_intensity.ply
```

Examples

We prove you with a few examples that show you how you can work with the data.

Example 1: Visualize PLY

This file shows you how you can read and display the pointclouds in python. This is useful so you don't always have to generate ply files and open them in Meshlab. Also it gives you access to the powerful Open3D library that has a ton of features to work with pointclouds. Please check it out here: http://www.open3d.org/

Open3D also renders the points on the graphics card and has the same problem as Meshlab with our large coordinates. Because of that we subtract again the center.

Also this example shows how you can export pointclouds as ply files.

Example 2: Project to Camera

This example shows you how you can project points in the pointclouds into the camera images. In this example we project a whole segment into the planar 2 camera.

For this to work we need to use the `Projector` class. When given a folder, it reads the pose file `reference.json` which contains the position and angles of the camera and corresponding JPG image. Then you just give this projector and index, which pose to use and you can project arbitrary points with colors.

To find the pose / frame you want you can look into `proj.poses`, to find a pose that is close to your projected points. The render function has a few interesting parameters you should check out. Also you will notice that the calibration is not 100% correct.

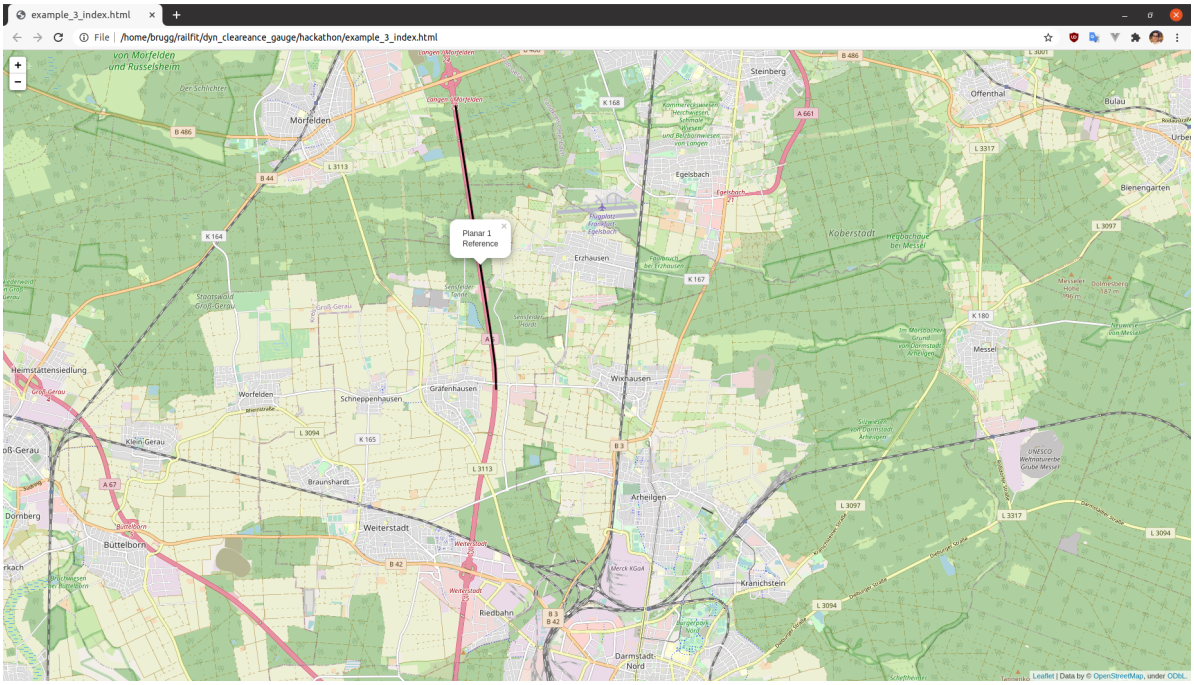
Also this example shows how to display 2d images.

Example 3: GPS Trajectory

In this example we show you how you can transform points in the pointcloud to GPS coordinates and draw them on Maps from OpenStreetMaps. In this case we draw the camera trajectory.

We provide two very simple functions that transform from and to GPS coordinates. And also we show how can generate interactive maps using the `folium` library.

The example produces an HTML that you should open in the browser.



Example 4: PointCloud Projections

This is the most advanced tutorial and it shows a very useful technique that is used when analyzing pointclouds. Namely creating and rendering 2D projections. What we do here is we want to project the segment along the driving direction on a 2D image plane.

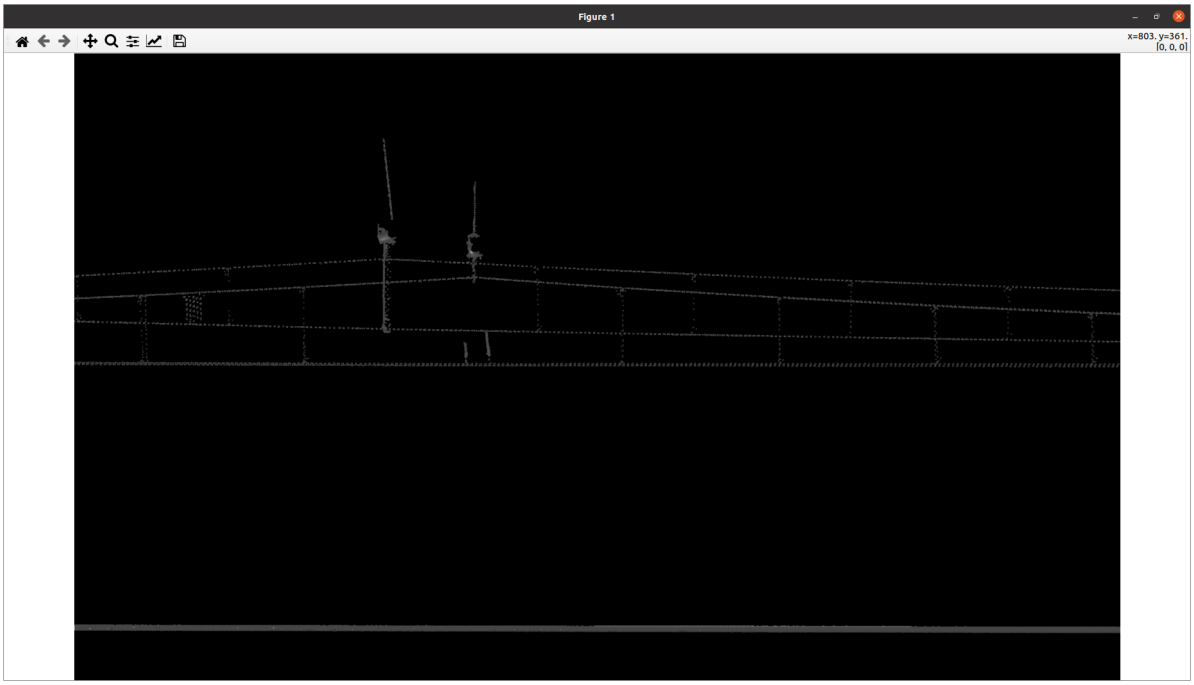
To do that we need to know the driving direction, which we get from the camera trajectory. We than find the points close to that vector with some vector calculations. After selecting them we project them on a 2d plane and scale them.

Finally we have provided you with a fast render function where you can render a list of 2D coordinates and intensities onto an image.

You don't need to make use of this technique, still we want you to know about it and give a basic example, if you want to go into that direction.

One advantage of the 2d projection is that one can use very basic image analysis to extract metrics / detect different things etc. And also one can usually very easily move between those two coordinate systems.

This is how such a projection looks like:



Final words

By now you should have been able to run all the examples. If not please reach out to us for help. You can further familiarize yourself with the dataset and read up on some common techniques regarding pointcloud handling.

And don't forget have fun with all of this. This contest is mainly for you to get inspired about this kind of work and to show it to others. So I hope this was already a good taste.

And in case you are totally overwhelmed with the complexity of the code and task we also want you to relax. The task is structured in such a way that it is not strictly necessary for you to write a bunch of code, but rather to get clever about pointcloud analysis in general. There are many ways this task can be solved.

This is just what we are familiar with and we wanted to provide you with a good start, in case you also like to approach this topic from this angle. Again enjoy.