LiDAR

Setup

Get started

- Install python 3.7.4 64 bit (you will need all your RAM), be sure to choose "add to path" when using Windows.
- Clone the erikrm/LiDAR git repository or copy the folder LiDAR.
- Install the needed python packages by running the following command inside the cloned folder:

```
pip install -r ./python_requirements.txt
```

- You are now ready to use the functions in the repository.

Tips and tricks

- To update the requirements list with the packages currently in use, run the following command in the working directory:

```
pip freeze > python_requirements.txt
```

read_pcap_from_file.py

This program reads in one pcap scan and the corresponding INS file from a flight. It then processes the pcap file, combines it with the INS data and makes las files containing the resulting data points. The program doesn't read in the INS .log file corresponding to the .txt with the timestamps. This has the effect that the number of leap seconds that existed at that date isn't taken in by the program. It is therefore very important to check that the correct leap second is used in the processing.

How too

To process pcap data follow these steps:

- Create a folder in a desired location with a name describing the data to be processed
- Copy the .xlsx settings file lidar_settings.xlsx to that folder
- Edit the copied version to describe your flight, the Settings sheet is the only sheet being processed by the programs, so you can use the rest to your discretion.

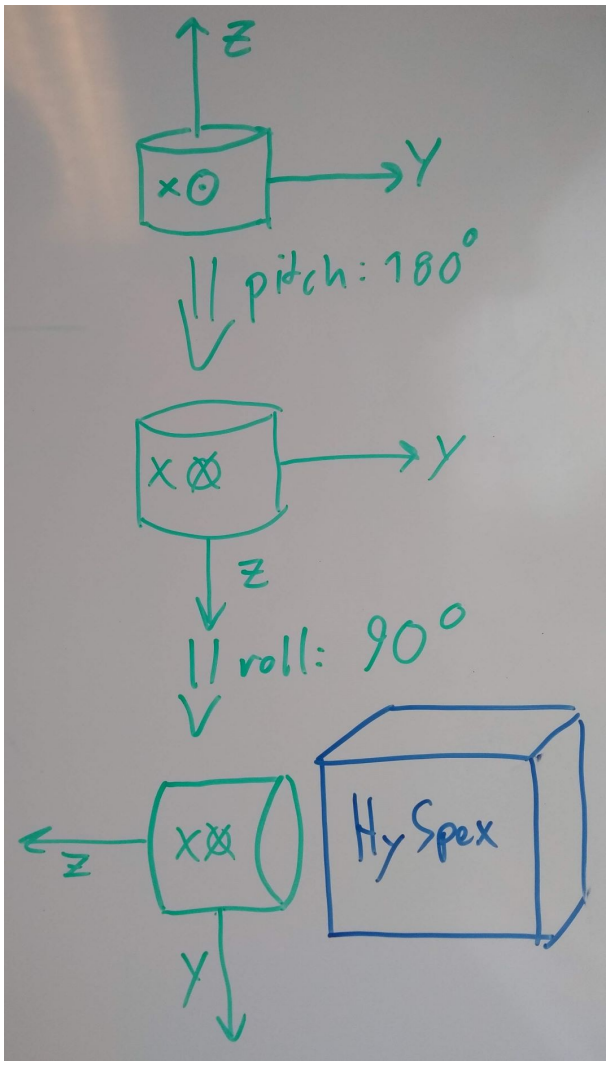
- Call read_pcap_from_file:

```
python read_pcap_from_file.py
```

- It will open up four file explorer windows after each other where you should choose:
  - The xlsx file containing your settings
  - The input pcap file (Containing the LiDAR measurements)
  - The corresponding INS file (They have to be in the same time span for there to be any output)
  - The out file directory

Excel file

The Excel file chosen when the program starts is where you configure every setting used during processing. Note that constants specific to VLP-32C is kept in the file lidar_values_and_settings.py and not in the excel sheet Settings.



Pitfalls

- Leap seconds, see read_pcap_from_file.

- Midnight, the timestamps given in the INS files are seconds since midnight between Saturday and Sunday. The timestamps given in the LiDAR measurements are seconds since the last hour change, top of hour (TOH). I choose to fix this discrepancy by using modulo on the gps timestamp to narrow it down to seconds since midnight on the current day. Then I used the NMEA string saved in the position packets sent by the LiDAR to find the hour of the day, I used this to convert from seconds since TOH to seconds since midnight (It is important to add the leap seconds here!). This might cause a problem if the drone for some reason is flown at midnight, maybe in the north in the summer. It might work too, I cannot test it.
- If packet_devisor (should be a natural number) is set to a number bigger than 1, every nth packet will be ignored. This means that six measurement will be plotted and then the program will skip n packets before reading six new measurements. This is an effective way of processing a huge field quickly, but it can lead to confusing patterns where it looks like the scan lines are parallel to the flight direction, when they are always perpendicular.
- RawPcapReader https://scapy.readthedocs.io/en/latest/api/scapy.utils.html?highlight=rawpcapreader#scapy.utils.RawPcapReader wants three inputs, I only give one. The magic and fdesc values shouldn't be a problem, it is something related to what kind of pcap format we are using. I think it works without since we are using the default protocol.
- When the program collects las frames into a more complete las file it is limited by the RAM on the computer. This is due to the open source program Laspy not having implemented append functionality to its file manipulation programs. My current solution is to divide the las files into batches. This is now not an automated solution, and the user have to define the number of frames per las file beforehand.

Tools

This folder contains all the functions that are not meant to be called directly, but rather used by the programs in the parent folder. The tools are divided into five scripts.

lidar_values_and_settings.py

This program contains all variables and setting used. If an Excel file is given with a sheet called "Settings", lidar_values_and_settings will use these values instead of its own default values. It will only parse values that are subject to change. I have not given the ability to parse constants so if a different LiDAR is being used, the constants must be changed inside this file. That is not a hard task, but I think it can be confusing for a normal user. This is the reason I parse from Excel, so that the user doesn't have to change any code during normal use.

div_functions.py

This file contains a diverse set of random functions.

write_to_las,py

This script contains all the code for reading and writing to las files.

udp_unpack

This file contains all the functions used in processing the LiDAR data. I would take a special look at rotation_matrix_array, interpolate_ins and transform_to_map as sources for error if the scans look weird. Especially note that interpolate_ins inverses the rotation from the ins. I had to do this to make it seem nice, but I'm not really confident about it.

serial_communication.py

This file contains the functions used for programming and communicating with the Garmin gps.