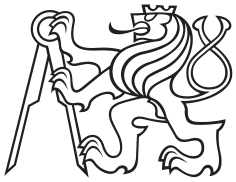
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Construction of 3D Point Clouds Using LiDAR Technology

Tomáš Trafina

Supervisor: Dipl. Ing. Milan Rollo, Ph.D.
Field of study: Cybernetics and Robotics
Subfield: System Control
May 2016

Acknowledgements

I would like to express my gratitude to my supervisor Milan Rollo for integrating me into the LiDAR Mapping project and giving me such useful comments, remarks and engagement through the research process of this thesis. Furthermore I would like to thank Martin Selecký for valuable advice on effective coding as well for the support on the way. Also, I like to thank my colleagues Jan Předota and Zuzana Tůmová as they completed my work by sharing theirs funcional sub-systems. Finally I appreciate Petr Váňa's help with libLAS integration into Windows OS.

Declaration

I do hereby declare, that I developed the submitted thesis independently, and I stated every piece of information sources which I used according to the methodical instructions about compliance of ethical principles during the development of a bachelors thesis.

In Prague, ...

.....
Signature

Abstract

The purpose of this work is to develop a system which establishes communication between LiDAR, IMU and GPS, makes necessary calculations over the acquired data and constructs 3D map using point cloud technology. Reduction of data imprecisions and uncertainty is also major objective of this thesis.

To verify the hardware was set up onto an UGV, later it will be moved onto an UAV. The vehicle was used for testing data capture right after the communication protocols and calculation algorithms were found reliable and fully functional. Eventually, computational algorithm's corrections and further sesnors calibrations were needed for precise outcome.

The system supports automation in measuring and mapping of landscapes and built-up areas. It significantly accelerates the process of creating usable data for related software applications, which are able to classify the point cloud's content, process it and therefore provide the user with relevant set of information about the scanned area.

Keywords: LiDAR, 3D, mapping, point, cloud, UAV, UGV

Supervisor: Dipl. Ing. Milan Rollo, Ph.D.
Dept. of Computer Science, FEE, CTU in Prague
Technická 2
16627 Praha 6

Abstrakt

Cílem této práce byl vývoj uceleného systému, který zajistí komunikaci mezi senzory LiDAR, IMU a GPS, následně provede nezbytné matematické operace nad změřenými hodnotami a zkonstruuje 3D mapu použitím technologie mračen bodů. Neopomenutelným cílem práce je také redukce nepřesností a chyb nad změřenými daty.

Pro ověření funkce byl použitý hardware instalován na UGV, v budoucnu bude přesunut na UAV. Po shledání funkčnosti komunikačních prokolů a výpočetních algoritmů, byly pomocí tohoto vozidla naměřeny testovací záznamy. Pro získání precizních výsledků bylo nakonec nutné analyzovat zkušební data a provést dodatečné kalibrace senzorů a korekce výpočtů.

Tento systém podporuje automatizaci v měření a mapování krajiny i zastavěných oblastí. Velmi urychluje proces vytváření datových podkladů pro softwarové aplikace, které jsou schopny rozpoznat obsah map, zpracovat ho a poskytnout tak koncovému uživateli relevantní soubor informací o prozkoumané oblasti.

Klíčová slova: LiDAR, 3D, mapování, mračna, bodů, UAV, UGV

Překlad názvu: Konstrukce 3D mračen bodů použitím technologie LiDAR

Contents

Figures

Tables

Chapter 1

Introduction

The usage of UAVs raises these days. With a hardware shrinking and increasing its computing power, we are able to mount more advanced systems on these vehicles and use them for automation of tasks wich would be very time-consuming when done by human. The motivation for this work was a development of a multi-purpose mapping system which would be capable of scanning terrain under trees.

This thesis introduces possible applications of an advanced sensor called LiDAR - Light Detection And Ranging, concretely the Velodyne VLP-16 LiDAR. This sensor unit is close relative to a generally known RaDAR (Radio Detection And Ranging) and uses the same principle for its function. However, light waves replace the radio ones. The objective is to synthesize useful point cloud file according to the selected LAS standard.

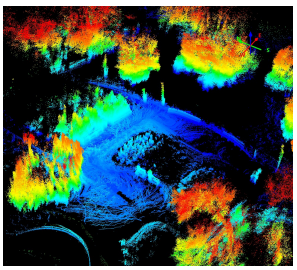


Figure 1.1: Charles Square in Prague

Initially, the project has started with ARM computer board and three sesnors: LiDAR, IMU (stands for Inertial Measurement Unit) and GPS (Global Positioning System) unit. The stated units were standalone and only had a function to gather outside world's data using theirs integrated sensors.

The goal was to combine these units into a working hardware/software system which would be capable of 3D point cloud's construction. Additionally, we wanted to identify every source of imprecission in the models so we can develop a fix immediately and therefore improve the mapping process to its best possible performance.

Firstly, it was necessary to set up communication protocols between the computer and each of the sensors. After that, we needed to develop mathematical algorithms which would be able to merge those measurements into the desired output. Some file format conversions were inevitable as the standardized output format was required.

Measurements and calculations were not precise enough immediately so we spent a lot of effort recalibrating the sensors and synchronizing the sets of acquired data. Along the way, we chose to encode the output in the ASPRS (American Society for Photogrammetry and Remote Sensing) laser (LAS) file format.

The structure of this thesis is the following. Firstly, we will cover the basics of the knowledge about a LiDAR focusing on those parts which are relevant for the map's construction. After that, we will discuss other sensors integration and usage of them along with the LiDAR for the map's creation describing each particular transformation. We will briefly discuss a usage of various vehicles. Thereafter, we will take a look at identified errors and imprecisions and theirs possible (used) corrections. In the end, we compare two LiDARs and uncover technology differences between LiDAR mapping and photogrammetry method in the terms of usage in various applications.

Part I

LiDAR and Its Usage

Chapter 2

LiDAR sensor

2.1 Introduction

For the very simplest LiDAR, we can imagine some sort of device for distance measuring using a laser beam. If we are able to mount this device on a rotating/swinging platform, we get a LiDAR with one firing plane (in the case of a rotating platform, there is a problem with the sensor's communication lines). There is a lot of various LiDAR types commercially available, with various numbers of firing planes and vertical/horizontal fields of view. A LiDAR definitely cannot substitute a radar, because it cannot reach such distancies. Contrariwise, it can substitute for methods which use photos/video records for model construction. Compared to them, LiDAR has far greater precision in objects surfaces recognition in the terms of scanning from greater distances (UAV applications). Combined with a camera, the system can deliver precise dimensions data along with objects colors derived from the camera's data.

Some of available LiDARs are able to measure distances up to 1,5 kilometers, those have a single firing plane. As the maximal range decreases, planes number usually raises, if we stay at the same financial level. There are two major utilizations of a LiDAR. It is widely used for terrain and building mapping, indoor/outdoor for various applications such as inspections and structures 3D plans. On the other hand, the system might process only the last scanned frame (one revolution/swing) and uses it for an environment recognition for a vehicle's autonomous navigation. These applications require LiDARs with as many firing planes as possible along with the largest field of view available (in other words, the frame needs to be in the highest resolution affordable).

2.2 Physical functionality

2.2.1 Medium

In contrast with a radar, a LiDAR applies light waves for the detection. Since systems are used in populated areas, they implement invisible laser's wavelength most often ranging from 700 to 1000 nanometers (according to the LiDARs on the market). Also, it must satisfy the class 1 (eye-safe) laser device criteria considering the wavelength and output power of the laser. For more, see the US-oriented (ANSI) Laser Safety Standard (ANSI Z136.1).

2.2.2 Time-of-flight

Firstly, we will cover the crucial principle of a LiDAR sensor (also used by radar). A LiDAR uses its light trasmitter, which sends out short pulse of a laser beam in one direction. The time of the laser shot and its direction are registered. The light wave travels through space until it hits some obstacle from which it reflects back. Considering the beam that heads back the same trajectory, just the opposite way, it finally reaches the LiDAR's light sensor, which registers the acquisition time and the power carried by the light wave (reperesents intensity of the light received).

When the process is done, the time-of-flight principle calculation takes place. The well known equation for velocity, distance and time dependence is used.

$$distance = \frac{acqusition\ time - shooting\ time}{2} \cdot speed\ of\ light \tag{2.1}$$

It is essential that the time difference must be divided by 2, because the light has traveled the measured distance twice.

2.2.3 Frame

In some cases, we can have just one fixed emitter/receiver pair in the device. These units are used for static measurements (the unit is still): as proximity sensors, in building construction industry etc. If we want to have some perspective of the surroundings, we need to mount this sensor on a motor and swing it from side to side or make it spin around. The last solution is difficult for relization due to an impossibility of simple wiring. You would probably have to use a wireless communication.

Nowadays, almost every bigger LiDAR unit has its inner sensors mounted on some form of moving platform, either swinging or rotating. In most cases, the unit does not even have just one emitter/receiver pair, but has more of them positioned vertically above and under each other (multiple firing planes) and therefore is able to "see" more. With these sensors mounted on a vehicle, in most cases, we don't have to deal with any movement of the unit

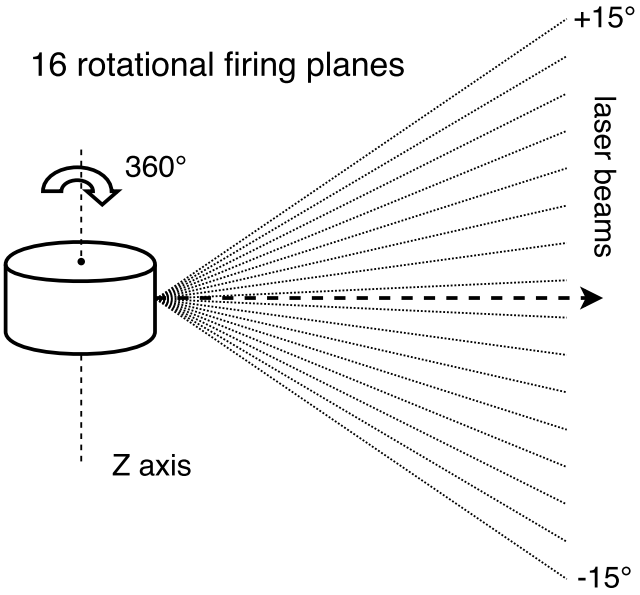


Figure 2.1: VLP-16 - field of view

itself, because the unit's field of view along with the vehicle's movement are sufficient enough for the map construction. See figure 2.1 and figure 2.2 for a couple of examples.

2.2.4 Multiple returns

Some LiDAR systems have an ability to register more than one return of a beam. As it was mentioned before, the light sensing part of the LiDAR measures the incoming light's strenght (energy). Advanced units record the light's energy continuously and then pick certain number of the most significant value peaks. See figure 2.3 for better understanding.

This principle is used, because we want to see the ground under trees and bottoms of waterworks and rivers. Generally speaking, we want to see through semi-transparent objects. When using the multiple return aproach, every single return represents some form of obstacle. If the scanning system, for example UAV, flies over a pond, first return will give the ponds surface distance, next one can be reflected by a plant growing at the bottom of the pond and the last one, which traveled through the previous two obstacles, hits the ground at the bottom. Similarly, if we fly over a forest, we can get two returns from two different leaves in the way and then the ground return.

Commercially available LiDARs can acquire from one to five returns per each laser shot. This maximal limitation is also given by the LAS standard (see chapter 6.1).

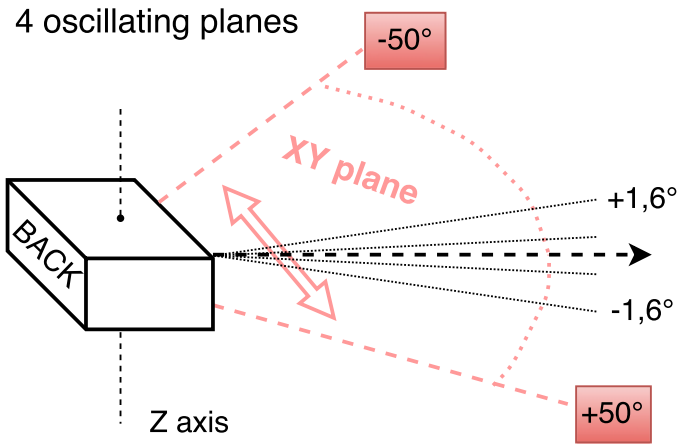


Figure 2.2: Ibeo LUX4 - field of view

2.2.5 Intensity

Some of the commercially available LiDARs have an ability to measure an intesity of a returned light beam, therefore we are able to distinguish a reflectivity of the scanned surface. This can be used for objects material recognition. Our work was to create a map with this information binded to every signel point leaving behind the processing of this parameter, so I do not cover the problematics in this thesis.

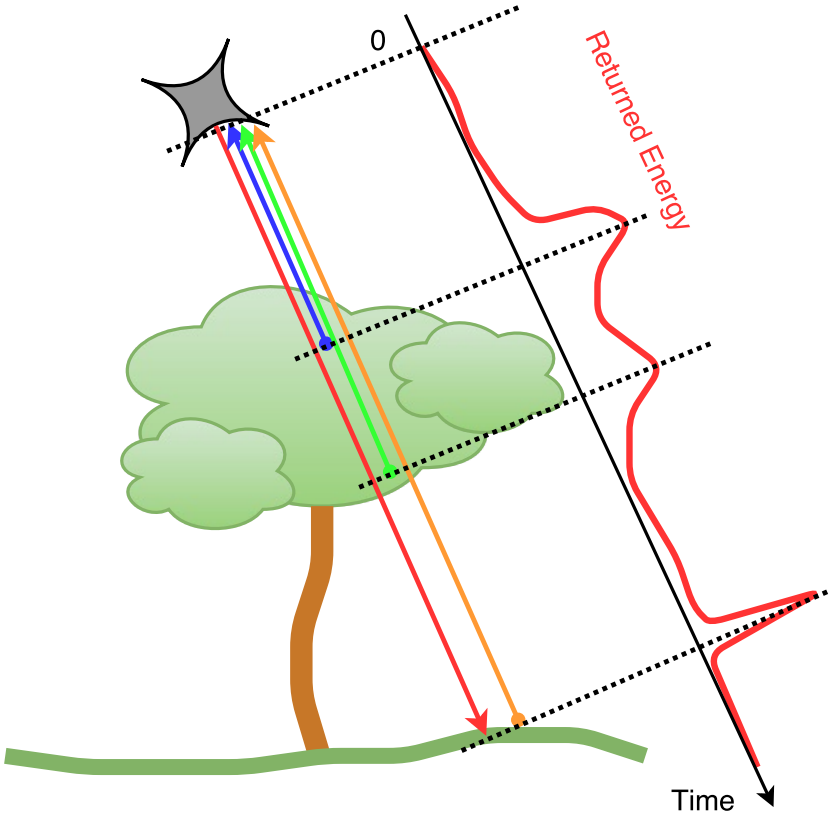


Figure 2.3: Multiple return principle

Chapter 3

Usage of the LiDAR

From now on, I will consider using the Velodyne VLP-16 LiDAR, which we have available for our experiments and analysis. Many similar parameters and functions of this sensor can be observed on other LiDAR's models as well. It is an intelligent sensor, which has its own internal processor and network interface. It provides the data through an ethernet line using UDP. Its settings can be changed with the use of HTTP web interface or by sending certain commands via TCP.

3.1 VLP-16 parameters

This sensor uses 16 vertical firing planes (emitter/receiver pairs) and is capable of acquiring up to two returns per one laser shot. It spins around its vertical axis and provides 360 degrees coverage in the horizontal field of view. Vertical field of view is 30 degrees, that implies 2 degrees spacing between each pair of adjoining firing planes. A relevant usable return can be up to 130 meters away (found experimentaly) and we get up to 300 000 scanned points per second. Horizontal resolution is variable and depends on rotation frequency (adjustable: 5-20 Hz). Received intesity value is between 0 and 255, for various intesity values meanings, see table 7.2, and for more details, see [Inc15]. You can see the VLP-16 LiDAR in figure 3.1.

Vertical resolution is given by the following equation:

$$vr = \frac{2\pi}{ppr} = \frac{2\pi \cdot f}{pps}, \tag{3.1}$$

where vr represents the vertical resolution in $rad/points$, ppr is the number of points per rotation in one firing plane, f is the rotation frequency (in Hz) and pps represents points per second in one firing plane.

3.2 Operating conditions

In this chapter, we will cover environmental restrictions for the LiDAR as well as its setting, data communication services and the output of the LiDAR's software subsystem.

Figure 3.1: The Velodyne VLP-16 [LiD15]

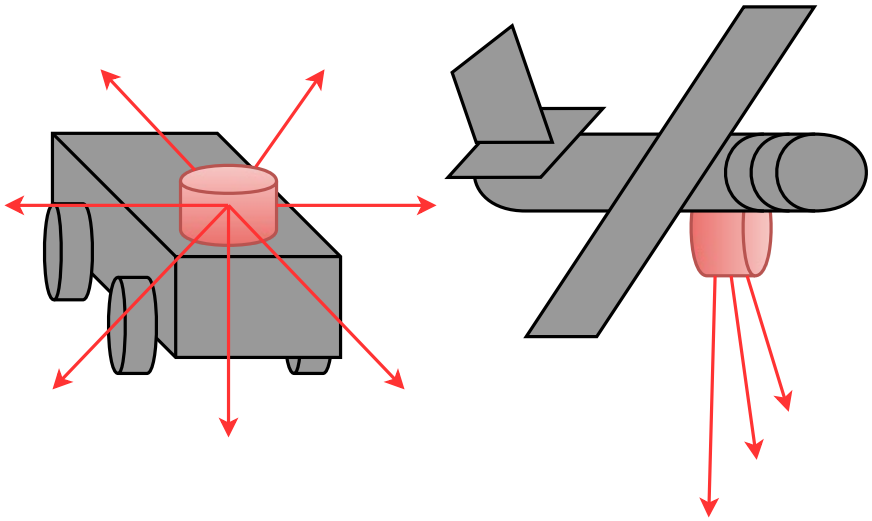


Figure 3.2: Mounting of a LiDAR

3.2.1 Sensors positioning

According to our needs for the application, we must think over the LiDAR's position and more importantly, its orientation. See figure 3.2.

On the left, we can see a horizontal mounting, which is widely used on ground vehicles, mostly the autonomous ones (the LiDAR is not used for map creation, it informs the vehicle about the environment so it can avoid abstacles and navigate itself through the track). In this setup, the sensor is able to scan all the way around and therefore, the system has great surveying ability to detect and classify objects and terrain right next to the vehicle. This has great usability in autonomous vehicles navigation through an unknown environment. One major disadvantage is that the LiDAR can shoot only under limited angle upwards resulting in tall objects not scanned.

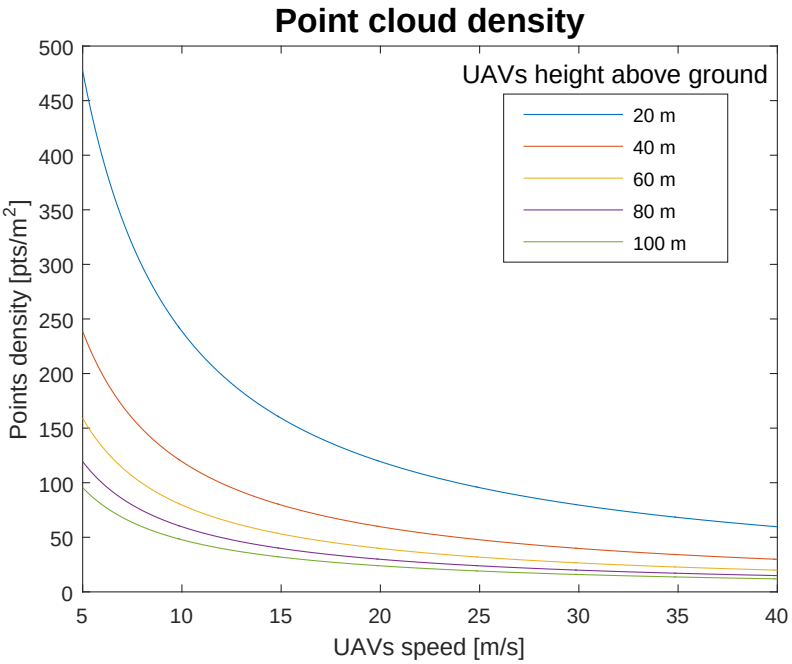


Figure 3.3: Point cloud's density

On the other side, we have a vertical mounting option. That is used on aerial vehicles for mapping in larger scale. Whole forests, plains and other predominantly nature objects and landscapes can be effectively scanned. Although we don't use the majority of the field of view, we can effectively take the advantage of the LiDARs range and fly relatively high above the ground.

The drawbacks of this approach are: approximately two thirds of the scanned points will be discarded. This fact implies decreased points/time ratio. We end up with lower accuracy due to the greater scanning distance and also, we get just DTM (digital terrain model), DSM (digital surface model) or both, because every object below is being scanned only from the top side. See figure 3.3. Compared to the horizontal mounting, where every nearby object is scanned from various sides as the vehicle passes it, here we have only rough idea about the object's sides shapes, because we can construct only the outline of an object and its top side's surface structure.

3.2.2 Weather

As it was described earlier, a LiDAR is able to detect transparent objects such as glass, leaves, foil, water etc. This becames a disadvantage in various weather conditions when the air is filled with some substance that can reflect/deflect the light beam. It can create ghost points, which do not represent any solid object present in the environment.

Field 4DEh		Field 4DFh	
Value	Meaning	Value	Meaning
37h	Strongest return	21h	HDL-32E
38h	Last return	22h	VLP-16
39h	Dual return		

Table 3.1: VLP-16: Factory bytes meanings

With the VLP-16 LiDAR, we have an option to switch between three return modes. We can choose to acquire the strongest return (the highest energy peak registered), the last return (the farthest detected object) or we can be given both of these. We can slightly eliminate the influence of the bad weather conditions by choosing the last return mode, because we have a chance, that the beam passes through the unclean air and finds its way to some relevant solid object. However, if the substance/precipitation is very dense, there is no chance for getting usable scans.

In our case, we can ignore this scenario as we do not fly a drone with unsealed electronics on board in this kind of conditions.

3.2.3 Settings

Additionally to the selectable **return mode** mentined earlier, the VLP-16 gives us an opportunity to set more parameters. We consider only those which are related to the points acquisition. We are able to set the **rotation frequency** in the range from 5 to 20 Hz. Moreover, we can acquire only a fraction of all scanned points. This allows us to discard unwanted points before they even leave the sensor saving the bandwidth of the line by setting **limited circular sector** to be scanned. For example: if the LiDAR has its angular origin facing down towards the ground, we can set this parameter to $\pm 60°$ and we end up with just a third of the LiDAR's horizon being send.

3.3 Communication

Every model of a LiDAR sensor has its own packet structure in which the data is being sent to the user. In case of Velodyne LiDARs, the information is effectively packed to save bandwidth of a communication line, as every point within a packet does not contain its full description and shares some parameters with other grouped points. According to the supplied packet's structure and timing information, we can retrieve every single record of a point without a precision loss.Packet's structure is shown in figure 3.4.

Factory Bytes - The last two bytes of a packet inform about the LiDAR's type and what return mode it is set to. Data can be decoded using the simple table 3.1.

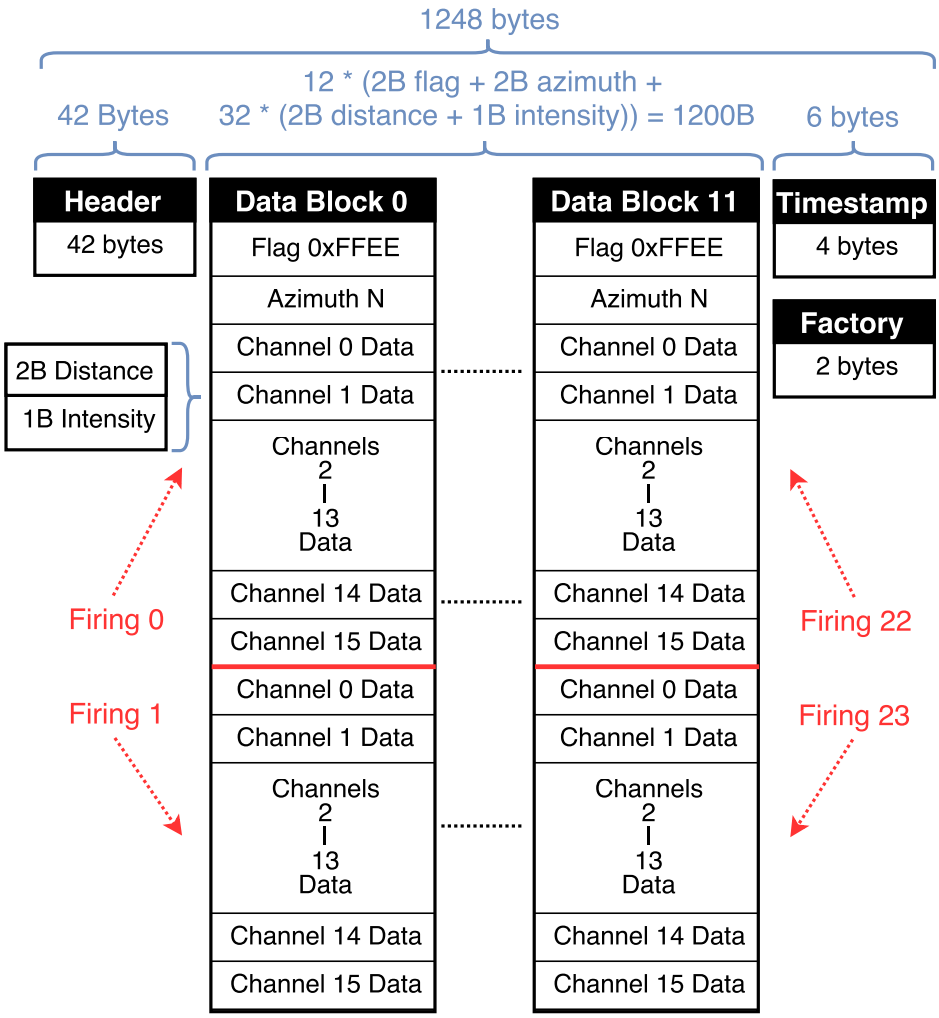


Figure 3.4: Packet's structure

Timestamp - Every single packet includes one and only one time stamp. It is represented by 4 bytes unsigned integer value and is written down from the 1200th byte in the payload. This time is the firing time of the first laser shot in a packet (microseconds since the LiDAR's startup/power-up). If we want to get a time stamp for every single point, we have to take the information about the lasers firings timing and do interpolation.

"All sixteen lasers are fired and recharged every $55,296\ \mu s$. The cycle time between the laser firings is $2,304\ \mu s$. There are 16 firings ($16 \times 2,304\ \mu s$) followed by a recharge period of $18,43\ \mu s$. Therefore, the timing cycle to fire and recharge all 16 lasers is $((16 \times 2,304\ \mu s) + (1 \times 18,43\ \mu s)) = 55,296\ \mu s$." [Inc15]

As we can see in figure 3.4, a packet includes 24 of these 16-laser firing groups (we will call these "firing block"), hence it takes $1,33\ ms$ to accumulate one data packet. This implies data rate of 754 data packets per second. This

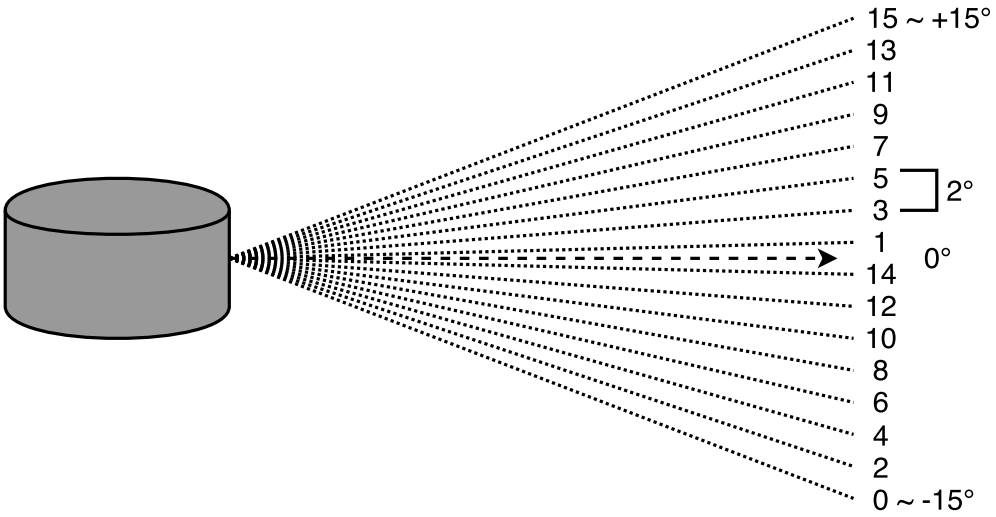


Figure 3.5: VLP-16 layers numbering

data rate doubles when the "dual return mode" is used, because the lasers fire at the same rate, but the sensor records twice as many laser returns (two returns from each laser firing).

Let us have a variable b which represents a firing block's number (0-23), and a variable p which holds an index of a laser firing in a firing block (0-15). We can now easily calculate relative time's offset (variable t_{offset}) between the first point in a packet and any other point in the same packet.

$$t_{offset} = (55,296\ \mu s \times b) + (2,304\ \mu s \times p) \tag{3.2}$$

To get absolute time stamp of the desired point, we have to add this relative time value to the packet's time stamp.

$$PointsTimeStamp = PacketsTimeStamp + t_{offset} \tag{3.3}$$

Vertical Angle - In figure 3.4, we can see that every single point's record has only distance and intesity measurements assigned to it. The vertical angle is based on a layer to which the point belongs derived from the record's row in a packet. We can get the idea of retrieving this information from the image 3.5.

Here is an example of simple retrieval of the vertical angle (channel corresponds to layer number):

```
if (channel % 2 != 0)
  verticalAngle = channel;
else
  verticalAngle = -15 + channel;
```

Horizontal Angle - We have a horizontal angle (we call it azimuth) recorded once for 32 points in total (once per 2 firing blocks). We call the set of two firing blocks a "data block". However, we do not have azimuths for even firing blocks. We have to do some simple interpolation here. Let us have one single packet, we want an azimuth of the second firing block. From the packet, we can retrieve azimuth of the first and the third firing block. We assume that the rotation speed between these two measurements is constant, so the desired azimuth must lay halfway between the two retrieved. We must check if the second azimuth rolled over from 359,99 degrees to 0 degree, and also use next packet for proper interpolation of the last firing block. Exemplary pseudo-code follows.

```
if (fBlock == 11)
  rawDiff = getAzimuth(nextPacket, 0)
                  - getAzimuth(thisPacket, 11);
else
  rawDiff = getAzimuth(thisPacket, fBlock)
                  - getAzimuth(thisPacket, fBlock+1);

azimuthDifference = 360 - ((360 - rawDiff) % 360);
```

Dual return mode differences - Previously stated parsing algorithms are used, when the strongest/last return mode is activated. With dual return mode activated, odd data blocks (understand, firing blocks 1,2,5,6,9,10) represent last returns. For each of them, the following data block holds strongest returns belonging to the previous last returns respectively (by rows). A packet now holds two returns for every laser shot, consequently just half as many laser firings as a packet in the single return mode, therefore the packet rate doubles as the LiDAR retains its firing frequency.

3.4 System implementation

Before describing the LiDAR's integration with other sensors and creating the desired point cloud, we must put the measurements into some easily manipulable coordinate system. For our needs, we want to transform the incoming coordinate system into Cartesian coordinate system with its axes aligned with the sensor's axes and origin in the LiDAR's mounting point, hereinafter called LMP (this is useful for assembly technicians when they measure mounting offsets between different hardware subsystems).

Coordinate system transformation - After proper interpolation work and conversions (discussed in chapter 3.3) on every point's measurement acquired from a packet, we have the following parameters available for every point: time stamp, distance, azimuth, vertical angle and intensity.

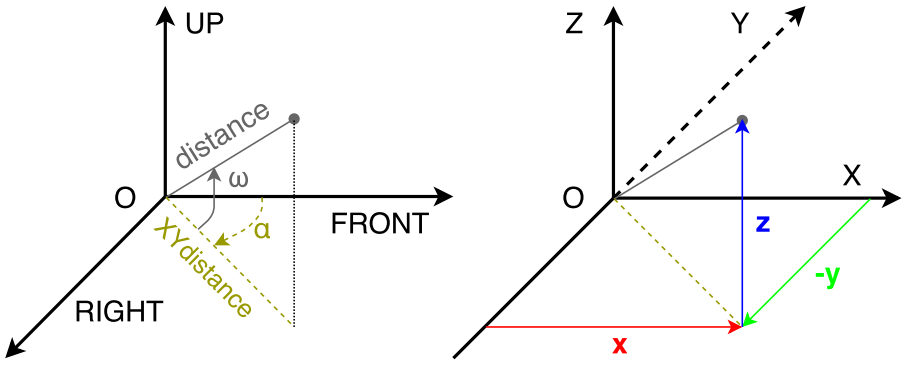


Figure 3.6: LiDARs coordinate system, desired Cartesian system

According to "Standard Units of Measure and Coordinate Conventions" [FP10], we decided to use Cartesian coordinate system (right-handed) with its origin in the LMP (screw hole in the center of the bottom side), so the X axis comes from the front of the LiDAR (negative X points towards the interface cable), Y axis points to the left. As seen from figure 3.6, the transformation is straight forward, except a couple of small corrections. We have to consider, that the emitter/receiver pairs are slightly off-centered, the distance measurement is reported with the origin in a receiver. However, we want the origin to be on the Z axis in the center of the sensor. Moreover, we don't even want it to be in the receiver's height above the bottom side of the LiDAR, so we also move it down by this offset. The following equations unreveal the whole calculation process.

$$z = d \cdot sin(\omega) + c_z \tag{3.4}$$
$$d_{xy} = d \cdot cos(\omega) + c_{xy} \tag{3.5}$$
$$x = d_{xy} \cdot cos(\alpha) \tag{3.6}$$
$$y = -d_{xy} \cdot sin(\alpha), \tag{3.7}$$

where x, y, z represent Cartesian coordinates of the point respectively to X,Y,Z axes, d is a distance between the LiDAR's emitter/receiver and a point, d_{xy} is the distance d projected to the XY plane (z = 0), c_{xy} represents the emitter/receiver off-center distance and c_z is the emitter/receiver height above the LiDAR's bottom side. Angles ω and α correspond to those in figure 3.6.

Part II

3D Point Cloud Creation

Chapter 4

Problem: Relative coordinates

When it comes to the point cloud map creation, we need to know exact position of each scanned point in the final coordinate system bounded to a local ground. To this point, we were only able to acquire coordinates relatively to a LiDAR's position. There are two different, widely used approaches that we can use to get referenced coordinates of the points (they are still relative to the position of a ground station). We can either use SLAM (Simultaneous Localization and Mapping) or integration of other sensor along with a lidar.

4.1 Our approach

There are a few well-known methods of building a 3D map. Each uses some sensor such as LiDAR, camera, ultra-sonic sensor etc. for an environment reception. For correct map construction, a knowledge of the scanning device's position and orientation is needed. SLAM is a favourite approach in the field of robotics. It is able to keep track of an agent's position in an environment while building a map of it. But, it requires high density of objects and at least partial enclosure of the environment. We do not use these algorithms, because they have problems in wide opened areas.[DWFIB06] This is why we chose an integration of an IMU and a GPS module instead (details are described further below).

For the first reason, the SLAM approach loses precision with an environment having low local density of obstacles. Second issue is its computing requirements. These algorithms must run in real time, on-board the vehicle. This requires high computing perfomance which is not achievable with the embedded computers (we use Toradex Colibri T30 for this project). Finally, our approach gives us a possibility to apply raw data acquisition followed by offline desktop processing. In next sections, we cover the sensors just briefly.

Figure 4.1: 3DM-GX4-45 [LLC14]

4.2 IMU

The IMU stands for Inertial Measurement Unit and is used to determine attitude (orientation) and sometimes position of a vehicle. We use 3DM-GX4-45 from Lord Microstrain (in the figure 4.1), which has a GPS module integrated. "The 3DM-GX4-45 utilizes the strengths of integrated multi-axis gyroscopes, accelerometers and magnetometers in combination with GPS, temperature and pressure readings to provide high accuracy position, velocity, attitude (including heading) and inertial measurements." [LOR14]

4.2.1 Kalman filter

As an extension, the device has an integrated processor which implements extended Kalman filter. It takes measurements from all available sensors and uses them to provide more precise data. This filter is classified as a dynamic as it has better outcome when the device is moving. The filter does not provide only better precision, moreover it is capable of making estimations of future states of the device. See the IMU's manual for more details. [LOR14]

When we want RPY (roll, pitch, yaw angles) information, the filter takes measurements from magnetometers and gyroscopes and combines them to provide more accurate data than from individual sensors. On the other hand, the GPS itself provides measured position only four times per second. The Kalman filter is able to estimate the device's position in between those measurements by calculating a velocity vector from accelerometers with respect to recorded orientation (from magnetometers and gyroscopes). RPY is reported in form of Euler angles, see figure 4.2, where α = yaw, β = roll, γ = pitch. Axis N points towards the north.

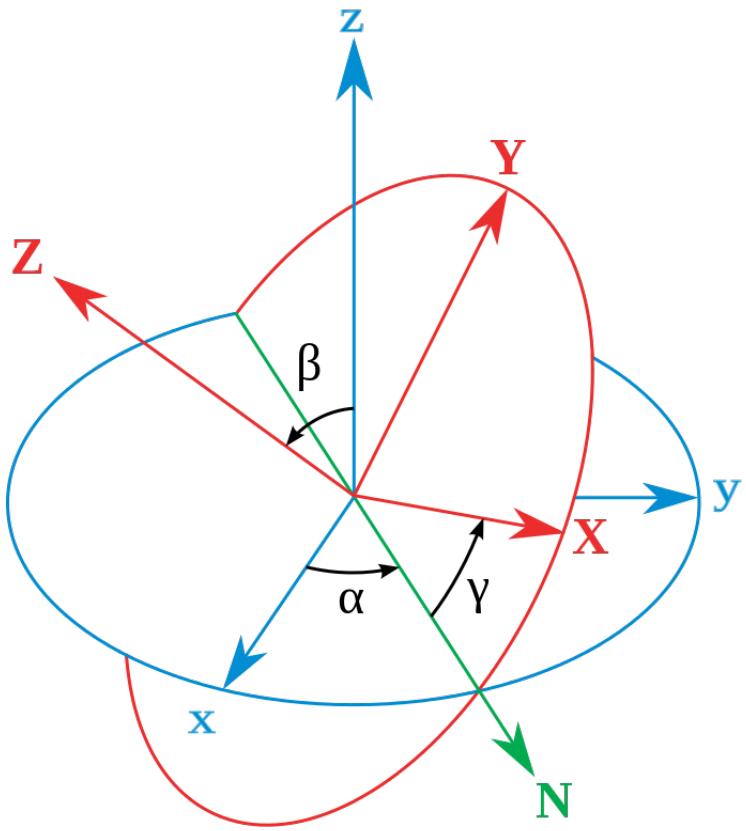


Figure 4.2: Euler angles [Bri08]

4.2.2 Output data

Because the GPS integrated in the IMU is not accurate at centimeter level (which we are aiming for), we use an external GPS described in the following section. The manufacturer provides a GUI (graphical user interface) in which we can save a desired configuration to the unit. Very important configuration option is what measurements the unit shall feed. As we chose only three angles to be reported, we were able to lower the baudrate of the communication and decrease the traffic of the computer's bus.

4.2.3 Usage

Mounting: The unit should be mounted flat onto a vehicle's horizontal plane. If we want it to be oriented in a different way, we have to save the angle deviations from flat orientation into the unit using dedicated GUI, so the unit can provide the measurements relatively to the correct plane.

Calibration: Magnetometers and gyroscopes require calibration. Gyroscopes are calibrated by simply placing the vehicle on a horizontal ground and leaving it still during a gyro bias capture. Magnetic calibration is a necessary step for proper IMU's functionality. The metal frame and wiring of the vehicle

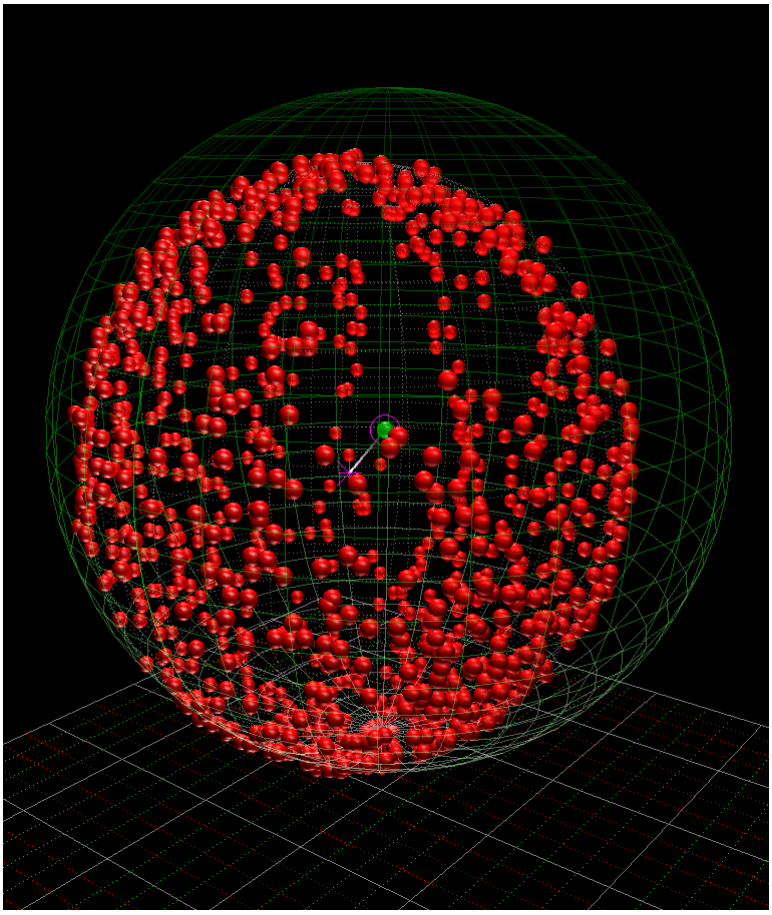


Figure 4.3: Calibrational elipsoid

bend Earth's magnetic field and cause incorrect IMU's measurements. The whole process of the calibration is described in the unit's user manual stated above.

4.3 GPS

For our needs, we want to use the most accurate GPS unit available. RTK (Real Time Kinematic) GPS is the best choice, as it provides centimeter-leveled accuracy.

4.3.1 Used device

We chose Piksi RTK GPS from Swift Navigation.[Swi13] RTK is a method used to improve the precision of position data derived from standard satellite-based positioning systems. It uses measurements of the phase of the signal's carrier wave relying on a single reference station to provide real-time corrections. As this system provides relative positions, it needs two RTK devices. The base station measures its position for some time, then takes the average as a

fix. Afterwards it uses differences between its measurements and the fixed position to provide the mobile device with real-time corrections (deviations caused by the atmosphere). More details can be found in [Tů16].

4.3.2 Planned device

The Piksi RTK looses satellites very quickly, when we pass an object which obstructs the view of the antenna at the sky, therefore is unable to reliably hold fixes. The planned replacement, Novatel OEM628, uses two frequencies for satelitte's data reception. It can provide a data with a frequency of 20 Hz (Piksi is capable of 10 Hz data provision) and with accuracy less than 5 cm. It does not need second paired device, because it uses static reference net (static GPS antennas which have precise location measured by surveyors). The biggest advantage is that it is more reliable in the terms of getting and holding the fix.[Tů16]

4.4 Interpolation

Let's assume that the communication lines have a negligible traffic delay and the processor of the on-board computer receives the data values almost immediately after they were measured by the sensors. The principle is the same for each of the three devices. Right after the data is read from the line by a dedicated method, the computer's current system time is assigned to the record. The record is then stored into a data buffer. Each sensor has its own buffer and a thread managing the sensor's services.

The frequencies of the three sensors data feed differ. To solve this problem, we must interpolate data from two of the sensors. The process goes as follows:

- We take a LiDAR's point from its buffer.
- We take first two IMU's records from its buffer.
- If the LiDARs point time stamp is lower number than the time stamp of the first IMU record, the point is discarded. This repeats until the point's time stamp is greater than the first IMU record's time stamp.
- When the point's time stamp lies between the two time stamps from IMU records, the two IMU records are used for the point's transformation (described below in 4.5).
- If the point's time value is greater than the second IMU record's time, the first IMU record is discarded, the second takes place as the first and the second place is filled with a new record from the buffer.

We described the principle on IMU records but the same applies to the GPS records in parallel.

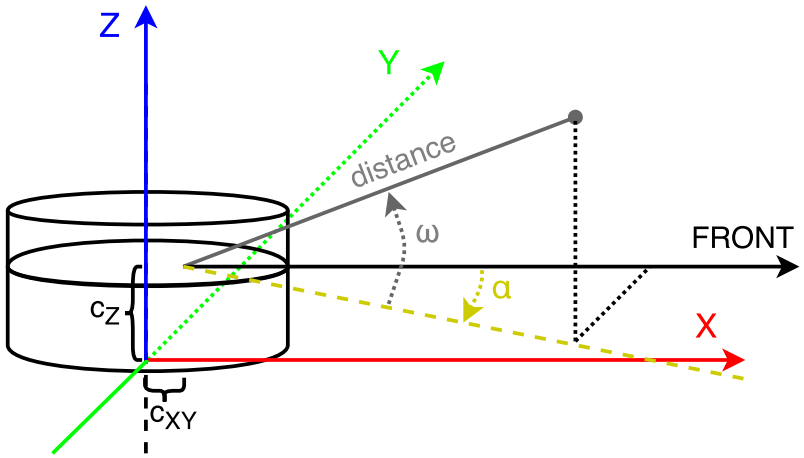


Figure 4.4: Ilustration of the first transformation

4.5 Transformations

Once we have records from the three sensors synchronized, we need to get every single point into its real position in a map. We split the transformation process into small, more synoptical sub-transformations, which can be done one after another.

Let us have one point, two IMU records surrounding the point with respect to its time stamp and two GPS records respecting the same rule. Now we are able to place the point into the final coordinate system.

4.5.1 Coordinate system transformation

This intersystem transformation along with the corrections to the LiDAR's mounting point was described earlier in section 3.4. After this transformation we have the point's position given in Cartesian coordinates relatively to the LiDAR's current orientation (axes are perpendicular with the coressponding LiDARs sides) and its mounting point as the origin. See figure 4.4 for better understanding.

4.5.2 LiDARs mounting angles

We want the point to be referenced in a coordinate system where the axes fulfill the following rules: X axis points forward, Y points to the left of the vehicle and Z axis points upward. To achieve this, we need to use a rotation matrix built up on the static mounting angles between the LiDAR and the vehicle. We use Euler angles for angles measurements, figure 4.2 shows proper Euler angles representing rotations along z, N, and Z axes consecutively. The xyz (original) system is shown in blue, the XYZ (rotated) system is shown in red. The axis N, which represents axis x after the first rotation, is shown

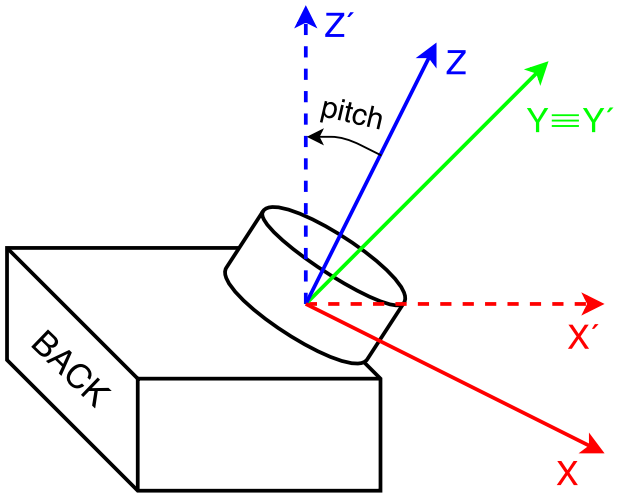


Figure 4.5: One of the LiDAR's alternative mountings

in green. In our case, the LiDAR is usually mounted horizontally or tilted only to one side (only one angle is non-zero). After the rotation we have the point referenced relatively to the LMP with respect to the orientation of the vehicle frame. In figure 4.5, an example with the LiDAR tilted forward by 45° is shown. The rotation matrix applied to the previous coordinates is:

$$\begin{bmatrix} cos\alpha \cdot cos\gamma & cos\alpha \cdot sin\gamma \cdot sin\beta - sin\alpha \cdot cos\beta & cos\alpha \cdot sin\gamma \cdot cos\beta + sin\alpha \cdot sin\beta \\ sin\alpha \cdot cos\gamma & cos\alpha \cdot sin\gamma \cdot sin\beta + cos\alpha \cdot cos\beta & cos\alpha \cdot sin\gamma \cdot cos\beta - cos\alpha \cdot sin\beta \\ -sin\gamma & cos\gamma \cdot sin\beta & cos\gamma \cdot cos\beta \end{bmatrix}.$$

4.5.3 Vehicle's orientation

Now, when we have the point referenced in the vehicle's coordinate system, we want the axes of the next coordinate system to be binded to a local ground. We use right-handed coordinate system with X axis pointing north and Z axis pointing up.

Alike the LiDAR's mounting angles transformation, we use the rotation matrix to apply the angles measured by the IMU, which are Euler angles too. The matrix is stated in subsection 4.5.2.

Because we do not have an IMU's measurement exatly binded to every point (by time), we use the two surrounding records. We take a point rotate it using the first rotation matrix and then rotate it using the second one. We end up with two positions of the point. Now we apply simple linear interpolation based on time. Transition between the coordinate systems is ilustrated in figure 4.6. The dashed system is binded to a local ground.

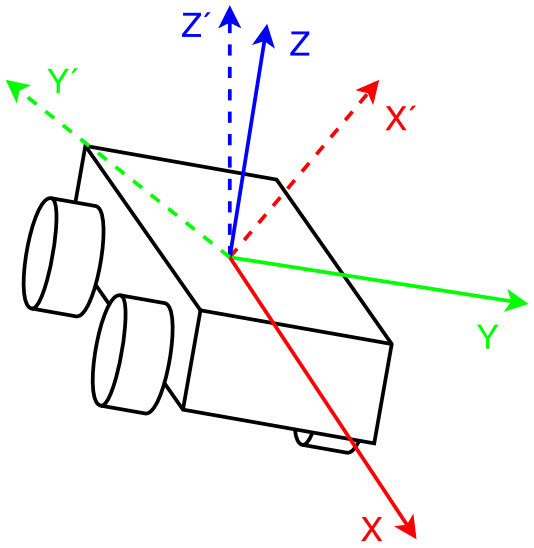


Figure 4.6: Vehicle tilted forward and to the left with heading aprox. 180°

We took an advantage of one feature integrated in the IMU's system. As with the LiDAR, the IMU may be mounted sideways or under variable angles. We cannot, do this mounting offsets fix by our software, because the IMU has specific roll, pitch, yaw angles representation based on various octants (quadrants). The pitch is reported only in range of $\pm\pi/2$, therefore if we mount the unit tilted forward by $pitch = \pi/2$, the unit will report the same values for the vehicle being tilted forward and backward (descending from the $\pi/2$ as the vehicle deviates from the horizontal plane to any side). See figure 4.7 for better understanding.

Because the IMU has its coordinate system situated upside-down (right-handed with X pointing forward and Z pointing down), we use the IMU's GUI to set the mounting angle roll to π, so the coordinate system corresponds to ours. That is because we mount it to the top side of the vehicle.

4.5.4 Vehicle's position

The final step is to reference the point to the ground station's position used as the origin of the desired map. This transformation is very trivial as it only moves the origin from the VRP (vehicle's reference point),at the moment it is at LMP, to the ground station's position by the relative coordinates provided by the RTK GPS. It is simple translational transformation, however, we must consider the fact, that the RTK GPS uses NED (north, east, down) coordinate system for y,x,z axes respectively. So before we apply the coordinates to our transformation, we do simple rotation of this system by interchanging the axes and theirs orientations, so it matches our coordinate system where we want X to point to the north and Z upwards. See figure 4.8.

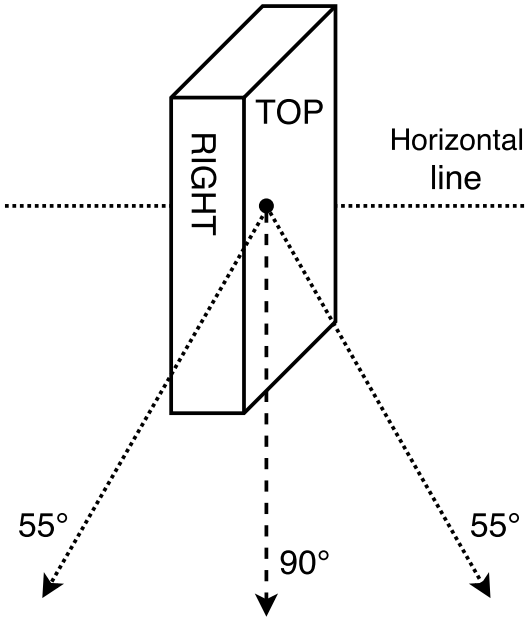


Figure 4.7: IMU's angle resolution issue (reported angles)

4.6 Outputs

We have two options for the calculation process. We can choose between on-board map construction or store raw data from the sensors with desktop post processing.

4.6.1 Calculated point cloud

When everything is optimized and the on-board computer can handle the computing requirements of the processing algorithms, we can run the calculation in real-time during a survey. Data from the sensors is merged and a map is constructed resulting in one binary file containing a point cloud and one CSV file with trajectory records (this is used for displaying the trajecotory in the model when viewed). The file is constructed using given parameters (using only fixed positions from GPS, ignoring points which are farther than X, ...) which cannot be reversed or changed after the file's creation (vehicle's survey in a terrain).

4.6.2 Raw sensors records

This approach stores raw data from each sensor separately resulting in three binary files as an output from the vehicle after a survey. A point cloud from it is calculated on a PC and allows for better debugging. It also gives us the possibility to generate models with various parameters from one data set. If we detect some major problem in the data set, it can be fixed.

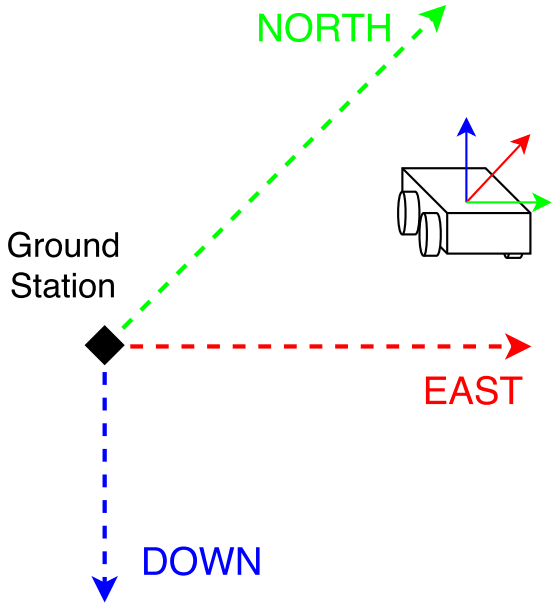


Figure 4.8: Relative position of the rover from the base

Chapter 5

System setup on a vehicle

5.1 Experimental UGV platform

UGV stands for unmanned ground vehicle. The most important fact is that the LiDAR is mounted horizontally (3.2.1). This means, that the vehicle is able to scan the terrain all the way around, but only shoots 15° upwards, so tall objects must be scanned from greater distance to register theirs top segments.

As the vehicle moves on the ground, we experienced many problems getting a GPS fix on locations with high objects surrounding the vehicle, therefore blocking a line of sight to the horizon. Regarding the mapping, there is no problem with trees or other semi-transparent object, but we are unable to scan tops of buildings and solid structures. Our UGV can be seen in figure 5.1.

5.2 Planned UAV platform

The future goal is to mount the LiDAR system onto a hexacopter called BRUS, pictured in figure 5.2. The whole hardware system, exluding GPS antennas, will be concentrated in one solid frame which will be dismountable from the UAV. GPS antennas will be mounted onto small elevated platforms above the vehicle.

This vehicle should have a big advantage in the terms of getting a GPS fix, because it has far more better view of the sky, more importantly, at the horizon (GPS needs to see satellites close to the horizon). The LiDAR will be mounted vertically, which implies smaller scanning area as roughly two thirds of the scanned points will be discarded (the laser beams point to the sky/UAV). However, the scanning distance is greatly increased, because it can get use of the maximal scanning distance of the sensor. On the other hand, we can choose to use decreased scanning radius when using the rover avoiding far scans.

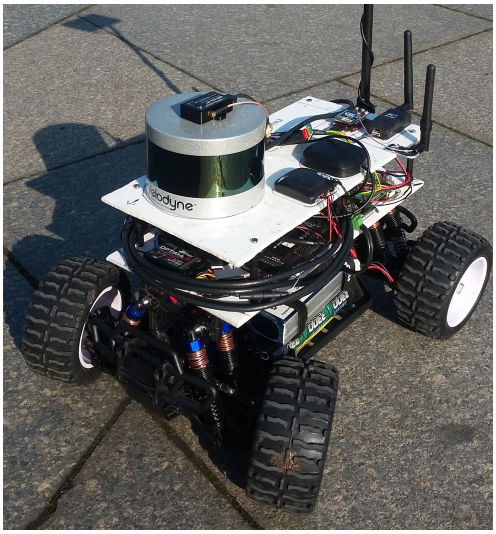


Figure 5.1: Testing UGV, the wheeled rover

Figure 5.2: Brus, the aerial drone

Last important difference to a UGV is that the UAV setup scans only terrain right beneath the sensor with $\pm 15°$ field of view, therefore it can be corrected only if the UAV flies over it again. On the other hand, the UGV scans every object several times from various positions, so its well covered and if a mistake happens (GPS loses fix, some data is lost), the object is still present in the final map, because it was scanned (will be scanned) from another position of the UGV.

Chapter 6

Offline conversion

6.1 LAS standard

As we try to make point cloud maps using a LiDAR, we decided to use the LAS (LASer) file format standard as there are plenty of viewing and editing software supporting it available. We use a las library called libLAS which is available under the terms of the BSD License on the website **www.liblas.org**

The library allows us to effectively store the point cloud maps and save memory space as it uses data compression. With the use of this format, the point's record may contain much more information than just its position, intesity and time. We can make use of classification flags (building, ground, vehicle, human, ...), return number, source (LiDAR, synthesized) and many other. More in the documentation on the website stated above. The library is only available for C, C++ and Python, therefore we had to write and compile Java methods wrapping the original ones for C.

6.2 Desktop converter

Desktop conversion is necessary as we do not have an output from the mobile system in LAS file format. Once a scanning mission in the field is done, we get the saved data from a SD card and pass it to a PC, where the conversions are made.

6.2.1 Raw data to map

This is the first step of the map creation if we used raw data acquisition in the field. As no calculations were done above the sensor's data during a UAV/UGV exploration, they must be done afterwards. Standalone Java application ensures this process by using the same calculation algorithms (even the same Java class implementation) as the on-board computer does using the real-time calculation. The output of this application is a binary file with a constructed map. Its format is the same as the outcome of the on-board computer when using the real-time calculation.

6.2.2 Binary map to LAS file

Second standalone application is used for las file creation. This one reads a binary file containing the final map and stores points into a las file using the libLAS.

We do not use libLAS right after the calculation process for two reasons:

1. The operation of a point being written into a las file requires more computing power, so we cannot use this aproach for the on-board computer.
2. For correct las file construction, we must first build up a header which should contain various parameters including point count and minimum/-maximum coordinate values. This requires the map to be processed twice (first run for the header parameters, second one for actual points recording). The header must be written to the file before the first point is stored. As the map construction process takes a long time, we use the binary file as a temporal storage of the computed map.

Part III

Imprecision and Uncertainty

Chapter 7

Imprecision and Uncertainty Problems

We can find various aspects of imprecision in this system. As we can't directly affect the perfomance of the hardware in the system, we firstly tried to choose the best hardware available according to our budget. Let's see table 7.1 for a list of the most important imprecision information of the chosen sensors, delivered by the developers and manufacturers. We want to discover every source of imprecision in the calculation and eliminate it or at least decrease it to a minimum, because we are trying to get maximal achievable accuracy of the model.

LiDAR VLP-16	
Beam divergence	3 mrad
Foootprint inaccuracy at 100 m	100 mm
Distance resolution	2 mm
IMU 3DM-GX4-45	
Roll & pitch accuracy	$\pm 0,25°$ RMS
Heading accuracy	$\pm 0,8°$ RMS
Attitude resolution	$< 0,01°$
RTK Piksi	
Position accuracy	centimeter-level
Position resolution	negligible

Table 7.1: Sensors accuracies

We set up various tests in the field to find the sources of imprecisions. Those included using straight/circle trajectories of the vehicle, scanning known object or moving the vehicle specifically around the ground station. For testing, we were strictly using raw data acquisition. Out tests took place at the top of a hill where the GPS had very good view of tke sky. At the end, after we removed all found errors, we did a complex ride on which we can show what accuracy we have reached.

7.1 LiDAR: Azimuth

7.1.1 Definition

Instead of the basic interpolation of a horizontal angle, that we have done in section 3.3, we can do a more precise one. After acquiring azimuth for every firing block, we ended up using the same information for 16 laser firings. However, these do not happen in the same time, therefore only the first firing has accurate azimuth information. As the others happen some time after, the emmiter/receiver array moves a bit and so has a different horizontal angle.

7.1.2 Solution

We assume that the rotation speed does not change between two data blocks in a packet. We already have azimuth for each firing block from the previous interpolation work. Now we will use the precise timing information of the lasers firings to calculate the real azimuths under which the points were taken. We apply basic linear interpolation respectively to a relative time of each laser firing in a firing block.

- Duration of the firing block: $t_{fb} = 55,296\mu s$
- Time between two adjacent laser firings in the same firing block: $t_{2f} = 2,304\mu s$

$$q = \frac{k \cdot t_{2f}}{t_{fb}} \tag{7.1}$$

$$a_k = a_b + [(a_{b+1} - a_b) \cdot q], \tag{7.2}$$

where a is an azimuth, b is an index of the firing block, k is an index of the firing plane and q is an auxiliary variable.

We look for inter-firing block and inter-packet interpolation as we have done before, as we also need interpolated value between the last firing of a packet and the first firing from the next packet.

7.1.3 Improvement

The rate of improvement depends on the firing plane from which the point originates, on the LiDAR's rotation speed and the distance of the point. The error value (between the simple interpolation and the upgraded one) can be quantified using the following relations.

Using the cosine equation:

$$d_e^2 = d^2 + d^2 - 2 \cdot d \cdot d \cdot cos(\alpha) \tag{7.3}$$

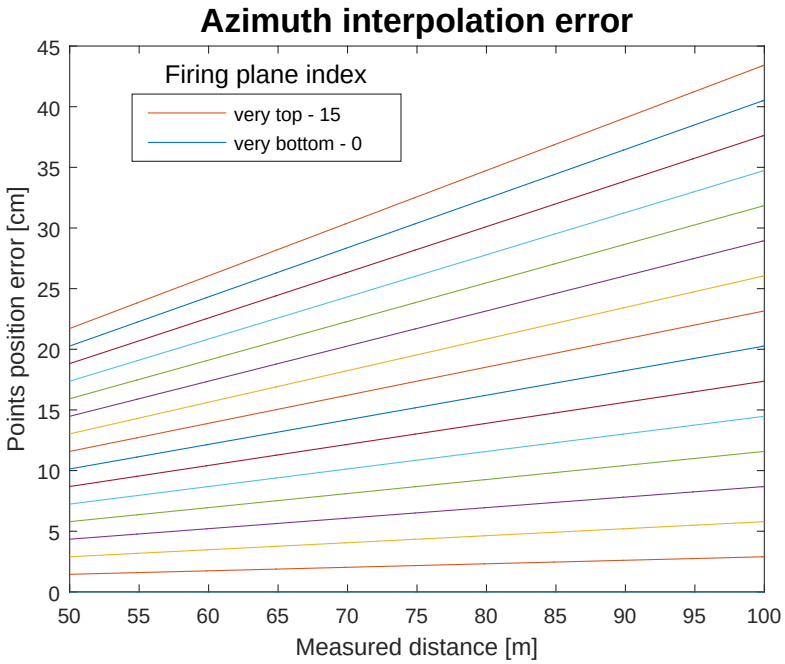


Figure 7.1: Error improvement (10 rpm)

$$d_e = d \cdot \sqrt{2 \cdot (1 - cos\alpha)}, \tag{7.4}$$

where d_e is the error position distance and d represents the distance to the point reported by a LiDAR. The angle α, which is the angle between the azimuth reported for the firing block and the real horizontal angle under which the point was scanned, is given by the following equation:

$$\alpha = 2\pi \cdot f \cdot (k \cdot t_{2f}), \tag{7.5}$$

where f is the LiDAR's rotational frequency. See figure 7.1 for an example of the error values when the LiDAR is rotating at 10 rpm. Other (not stated) distancies can be easily calculated from the diagram, as all functions are linear.

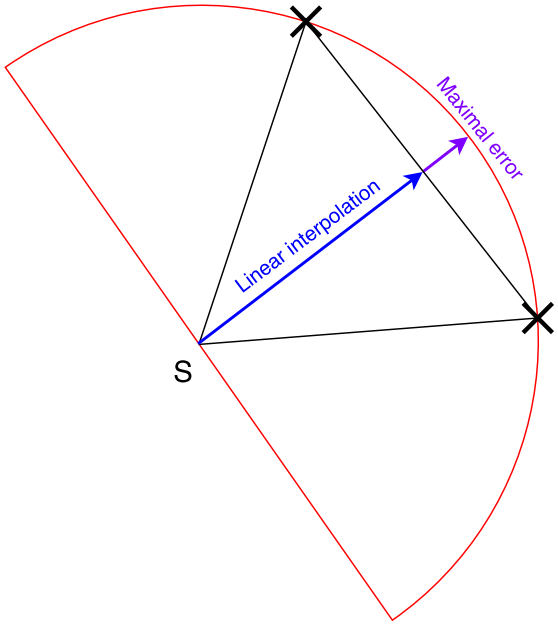


Figure 7.2: Comparison of arc and linear interpolation

7.2 IMU: Interpolation

7.2.1 Definition

As mentioned in subsection 4.5.3, we use linear interpolation of the IMU records. Let's say that we have two IMU records and one LiDAR point which has its time stamp half-way between the two IMU records time stamps. We take the point, rotate it by using a rotation matrix from the first IMU record, then rotate it by using the second record. In the end we take these two rotated points and define the final point as the one on the line half-way between the two rotated. Similarly, we use this principle with other ratios than 1/2. As we can see in the figure 7.2, this aproach does not preserve the measured distance of the point.

7.2.2 Solution

We would like the distance to be preserved. This can be achieved by using interpolation over an arc. The difference to the previous one is simple. We take two angle measurements of the IMU, use linear interpolation onto the angle values respectively to the IMU records and the point's time stamps. Afterwards we calculate a rotation matrix based on the interpolated angles and use it to rotate the point.

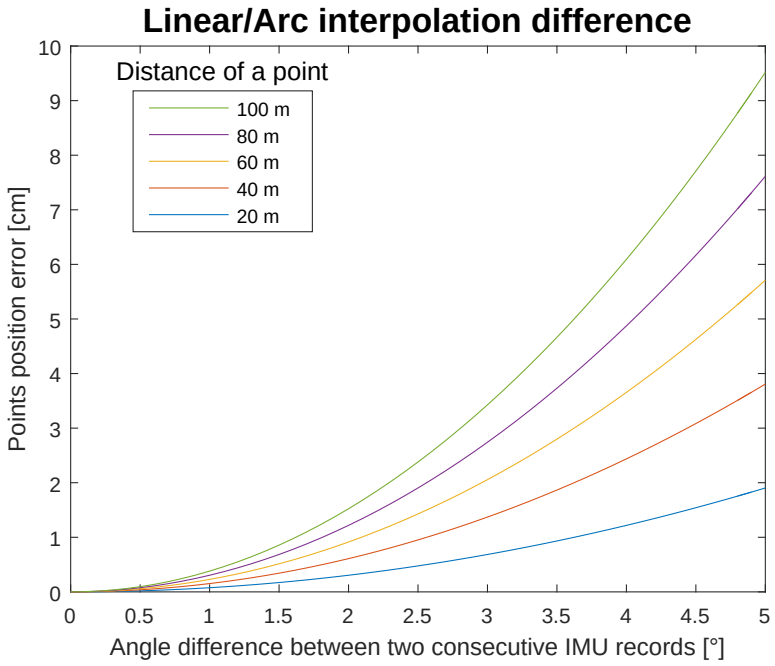


Figure 7.3: Linear/Arc interpolation difference

7.2.3 Improvement

Following equation quantifies the error value caused by linear interpolation.

$$e = d - x = d - (d \cdot cos(\frac{\alpha}{2})) = d \cdot (1 - cos(\frac{\alpha}{2})), \tag{7.6}$$

where e is the error value representing difference between positions of a point calculated by linear and arc interpolation. d is the distance reported by a LiDAR, x is the incorrect interpolated distance and α is an angle difference between two consecutive IMU records (between which is the point located, by time).

When the UGV falls with one pair of wheels from some edge, it can attain higher angular velocity (also with any kind of unpredictable vibration). We estimated, that it can reach a little over $2\pi/s$. Counting with IMU's frequency of 100 Hz, the angle difference between two records can vary from almost 0 to around 5 degrees. You can see possible error values in figure 7.3.

Actually, we do not use this upgrade, because the error of the original approach is negligible at smaller distancies (we can cut down the scanning radius) and the accurate algorithm takes a lot of computing power, as it calculates a rotation matrix for every LiDAR point scanned (aprox. 300 000 times per second) rather than hundred times per second.

7.3 System: Time stamps

7.3.1 Definition

This topic covers the most crucial methods of removing inaccuracies caused by different sensors data merging. If we want to merge data from different systems, we have to assign each record a time stamp from the same time system. (The GPS and the IMU use GPS time system, the computer uses operating system's time and the LiDAR uses its own time stamps.) Then we are able to pair the data from the sensors and make interpolation between the records.

The first most naive aproach is to assign the record with the computer's system time right when it arrives at the computer's port. However, due to a delay on the hardware line and a variable delay with which the processor registers the information, we can end up with huge (tens to hundreds of milliseconds) time difference between the data acquisition (by a sensor) and the registration (by a computer).

Example: The LiDAR scans a data packet at time 0 ms, it sends it through the line and the computer registers it at time 100 ms. In parallel, the IMU registers data1 at time 0 ms and data2 at time 50 ms, the computer acquires the information at times 50 ms and 100 ms respectively. All time information is the computer's time system. The merging algorithm will merge the LiDAR's data packet with data2 from the IMU according to the system time. But the LiDAR's packet should be merged with data1. This causes a model to rotate locally, when the vehicle is changing its orientation.

7.3.2 Solution

We fix this issue in two steps. Firstly, we measured the static traffic delay on the line by sending a packet with request to acknowledge its reception. The program records the time when the packet was sent and afterwards, records a second time when the acknowledgement from the sensor is received. We repeat this several times to ensure that one of the exchanges took place with almost none time delay on processors (in the device and in the computer). We take the smallest value. Therefore, we can assume that the difference between the two times indicates the time needed to pass data through the line (twice). This also includes bus services and other I/O operations done on both sides of the line (these happen every time some data goes through). Once the line is analysed, the value is written to the scanning program as a constant.

Secondly, before a scanning precedure, we do what we call a "time synchronization". The program listens to a sensor and look for the smallest difference between a packet's time stamp and its acquisition time recorded by the computer. After some time, it takes the value and substracts the static

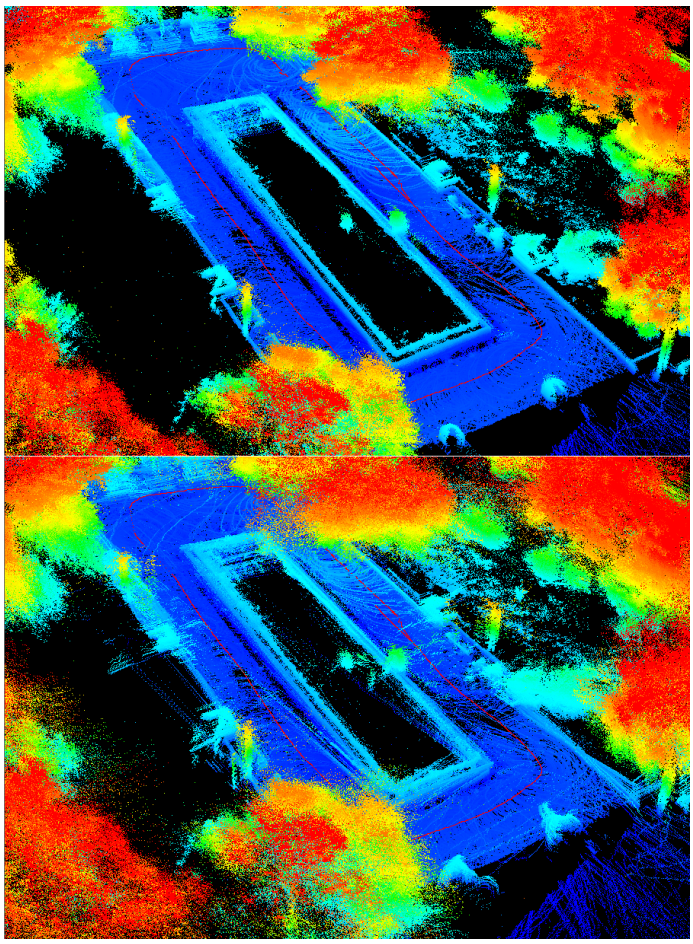


Figure 7.4: Timing issue: models comparison

delay on the line. Now we have a value which represents the shift between the two time systems. From now on, every time the computer receives a packet, it takes the packet's time stamp and adds the time shift value to it, so it gets the time of the data acqusition by the sensor in the computer's time system. This happens with the LiDAR and the IMU. RTK has the same time system (GPS time) as the IMU, so we apply the same time shift value to it.

7.3.3 Improvement

The strenght of this improvement is hardly expressible in numbers. It extremely depends on the vehicle's behaviour and speed during movement, as well as on the distance currently measured. In our case, the difference between the times in registration of the data that should belong to each other (have the same acquisition time) is in hundreds of milliseconds, so the improvement is clearly eye visible by the final 3D model. See figure 7.4 for a comparison. As we found this approach to be imprecise, we are going to add a GPS module to the LiDAR. The VLP-16 is able to receive NMEA messages and use theirs time stamps for its packets.

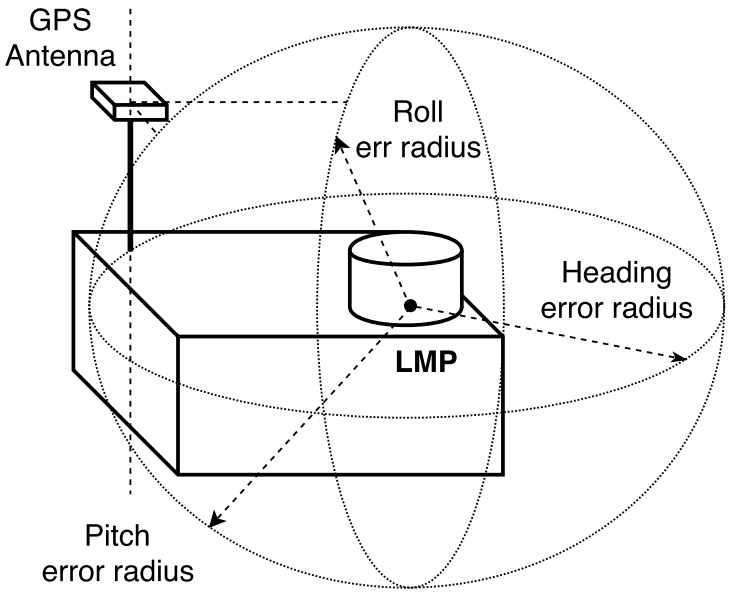


Figure 7.5: Orbiting GPS Antenna

7.4 GPS: Antenna offset

7.4.1 Definition

Like it was said in section 4.5, we use a GPS information to do the translational transformation of rotated points from previous subsystem (LiDAR and IMU). We assume that relative GPS measurements are done between the ground station and the LMP while they are done between the ground station and GPS antenna.

If we make a full circle with the vehicle (in XY plane) and the LiDAR is scanning some object the whole time, this object will drift around in a circle, with radius equal to the distance between the antenna and the LiDAR, in the final map. Imagine that the LiDAR stays still relatively to the coordinate systems origin, but the vehicle can rotate around it (this is an abstract case, but it may be realized, when the vehicle goes through the same place repeatedly from various directions). Now the relative coordinates of some scanned object are being the same all the time, but as the antenna on the vehicle rotates around the LiDAR, it provides various position data, so the object is rendered multiple times on various positions, see the picture 7.5 for better understanding.

7.4.2 Solution

The correction of this error is very straightforward. Before we use the measurements from the GPS, we must ensure, that they represent relative position between the ground station and the LMP (not the antenna on the

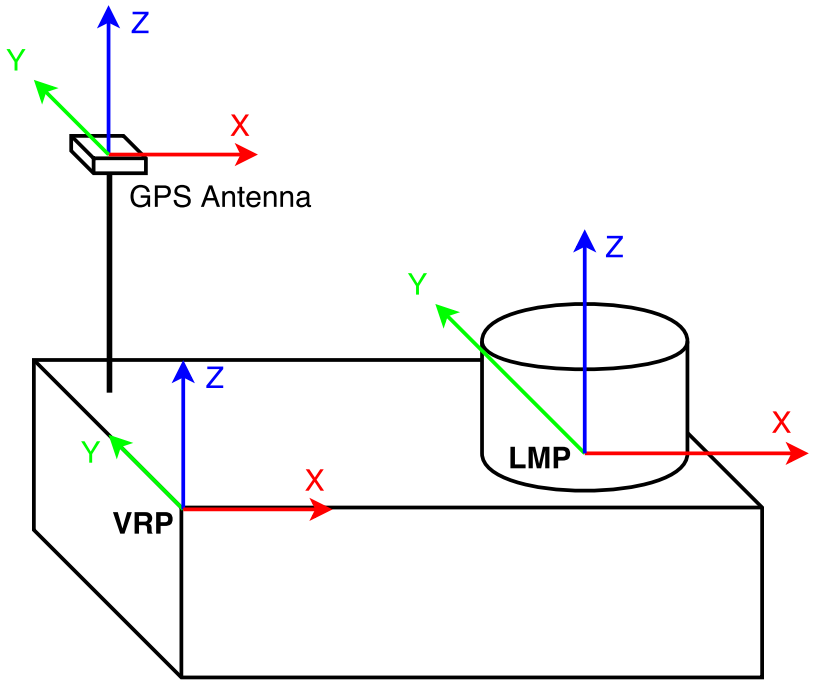


Figure 7.6: Translational transformation

vehicle). But we cannot just take the antenna's offset from the LMP and add it to the GPS provided coordinates. That is because, even if we know the position of the antenna, we cannot be sure, which direction from it is the LiDAR located (the vehicle can be oriented under variable angles). To include the angles information we have to utilize the IMU data.

As we mentioned in section 4.5, we have a chain of transformations done serially. To fix this issue, we need to add another partial translational transformation right after the second one (we have points referenced relatively to the LMP with system's axes binded to the vehicle's frame). Mathematically, we want to reference the point relatively to the antenna's position (instead of the LMP). Therefore we need one more translational transformation. We split this translation into two steps, because of facilitation of mounting offsets measurements during assembly. Firstly, we move the origin of the system into the VRP (vehicle's reference point). Afterwards, we move it to the antenna's position. See figure 7.6 for better understanding.

7.4.3 Improvement

The difference before and after the error's fix is very significant. It depends on the distance between the GPS antenna and the LMP. After the correction, objects no longer drift in circles.

7.5 LiDAR: Beam reflection

7.5.1 Definition

The technology of the LiDAR scanning is limited and we have discovered one disadvantage of the Velodyne VLP-16 while scanning various terrain structures. The LiDAR should capture non-reflective, semi-reflective and retro-reflective surfaces according to the following table.

VLP-16 Intensity reports	
Type of the surface	Returned value
Black, absorbent diffuse reflector	0
White, reflective diffuse reflector	100
Retro-reflector covered with semi-transparent white surface	101
Retro-reflector without any coverage	255

Table 7.2: Intensity reports based on a surface type

As we discovered, the retro-reflective materials are not sensed correctly. The principle of the LiDAR includes the light beam to reflect in the direction of the LiDAR with its remaining energy. We encountered a stone wall which had rounded and polished edge on the top. Therefore, when the beam hit the edge, it reflected in some random direction, then hit some object and returned the same way. As a result, the LiDAR recorder longer distance to the "wall" than it actually was (because the distance reported was the distance to the unknown object with a waypoint at the edge of the wall). In the end, we have ghost points behind actual obstacles. See figure 7.7 for an example.

7.5.2 Solution

This issue could be fixed by filtering outliers for example. The post-processing and filtering algorithms are outside the scope of this work.

7.5.3 Improvement

If a good filtering algorithm is applied onto a final point cloud, it should result in removing almost all ghost points while preserving relevant data of real objects in the environment.

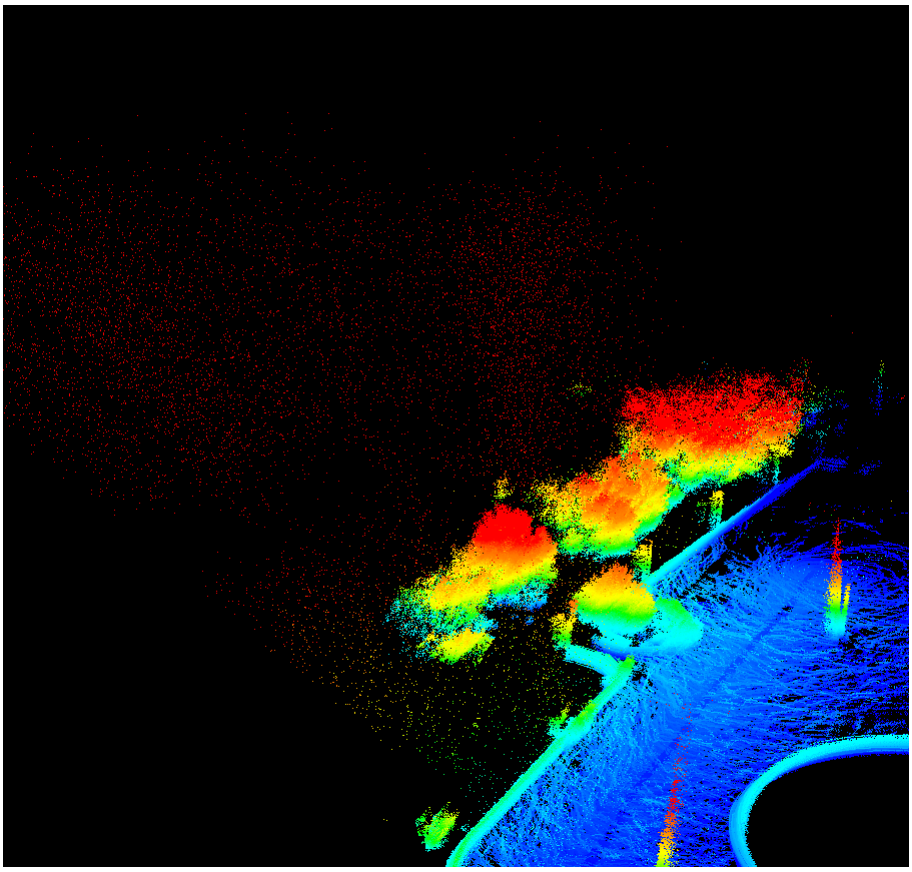


Figure 7.7: Ghost points

7.6 IMU and GPS Misalignment

7.6.1 Definition

It showed up, that the IMU provides bad heading records. Despite the high Kalman filter's precision, it does not provide heading with correct north/south orientation, because it can only measure the angle relatively to the default input value which is recorder wrongly, because the IMU uses only one sensor for its acquisition and it can be disturbed by environment. A point is scanned by the LiDAR and rotated using an IMU record. The coordinate system the IMU is using should be parallel (with all three axes) to the final one (GPS coordinate system), but it is not.

Situation is ilustrated in figure 7.8. The vehicle is facing north-west and standing directly to the east from the ground station, the GPS provides this information and it is correct. The LiDAR scans some object with azimuth zero, which implies that it is located to the north-west of the vehicle (to the north of the base), but because the IMU has captured bad initial heading, it reports the heading of the vehicle to be NORTH. The point was scanned directly in front of the LiDAR (front side), associated with the IMU information, it is rendered to the north of the vehicle. After moving by GPS value to the east, it ends up with the point being rendered to the north-east of the base (instead of north).

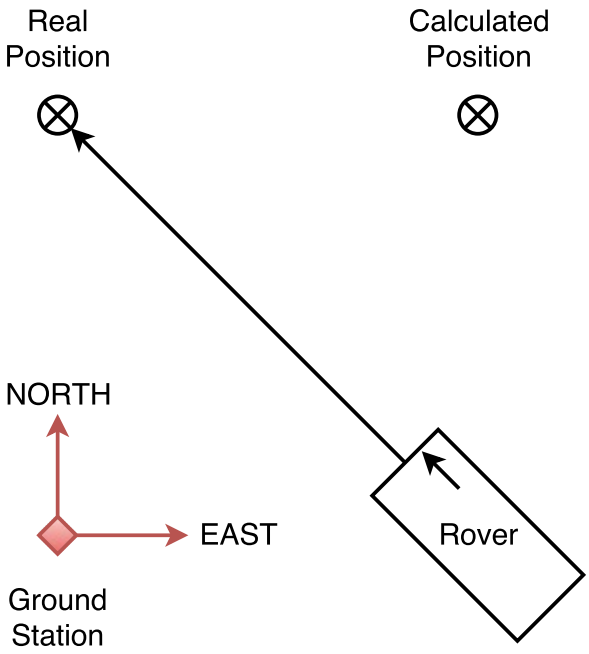


Figure 7.8: Calculation example

7.6.2 Solution

Current temporary solution is that we fix the models manually. We find some clearly defined geometric object and we find the error angle through it. We are able to apply this only on raw data sets. This solution is imprecise and we are working on an automated detection of the error angle. One of our ideas is to make the vehicle travel forward for some time and then, construct a velocity vector from the GPS information and compare it to the data from the IMU. The vehicle must move in the direction where the IMU has its X axis.

7.6.3 Improvement

It is extremely visible in the final map. See figure 7.9. Fixing this problem realizes correct implementation of previously intended operations and provides us with the desired result. The mistake will fully disapear with future automatic fixing algorithm.

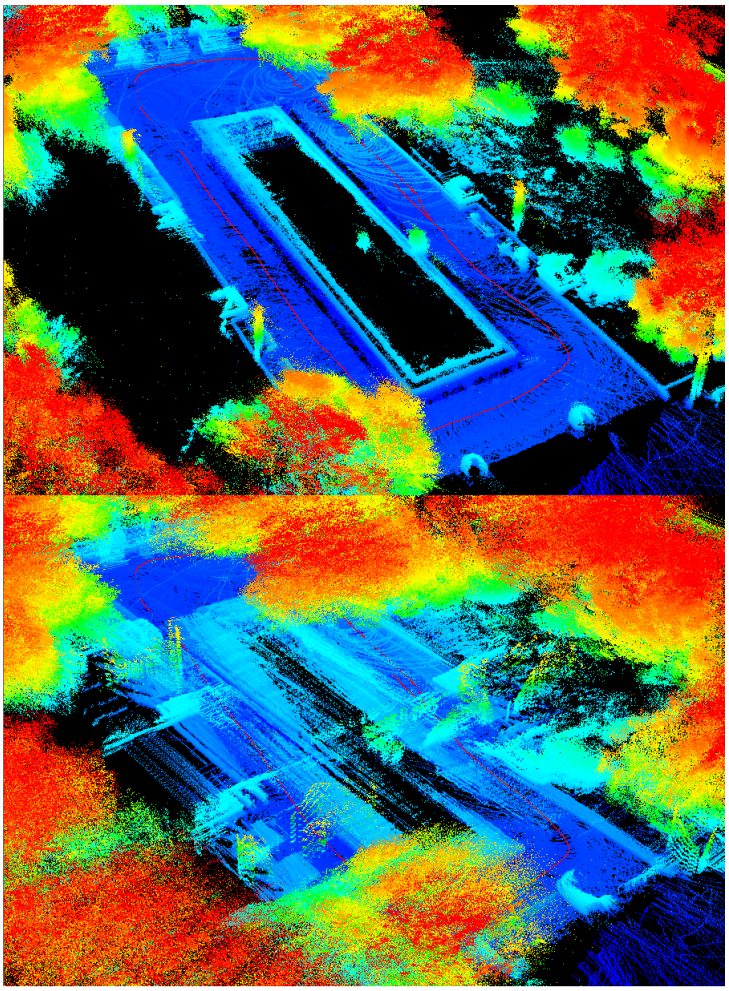


Figure 7.9: Misalignment: models comparison

Part IV

Technology Comparison

Chapter 8

Technology Comparison

In this part, we will introduce another LiDAR, which we have available, to show which differences can exist between various versions of a LiDAR. Also, we will describe another method to construct 3D models. Subsequently we will compare the models acquired by the LiDARs and the models from a LiDAR system and a photogrammetry method. We chose the photogrammetry method, because it is widely used for outdoor objects and structures modeling, as well as a LiDAR system is.

8.1 Ibeo LUX 4

Figure 8.1: Ibeo LUX 4 [GMb08]

8.1.1 Descripiton

These LiDARs were designed for use in automotive industry1. If used in front of a car, the processor inside is capable of objects classification. It can filter out points representing ground plane (road) and inform the car about important objects like cars or guardrail. When mounted on sides or the back of a vehicle, LiDARs can assist with parking and collision avoidance.

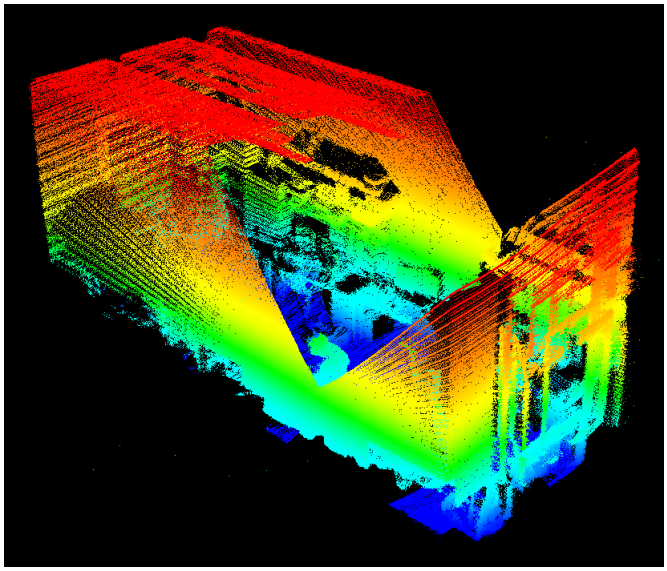


Figure 8.2: VLP-16: Scanned room

This LiDAR has four firing planes, but only two emitting units. Optics are used to split the laser beam. One emmited beam becomes two and their returns are registered by two separate receivers. Spacing between two adjacent planes is 0,8°(that implies 3,2°of vertical field of view). The LiDAR is able to sense a return from up to 200 meters and has a maximal measurement error of 10 cm (Velodyne 100 m / 3 cm). Horizontal field of view is more complex, it is not fully covered with all 4 planes (lateral areas are always covered just by two planes). Working area with 4 planes is 85°, adding expansion with 2 planes makes it 100°. For details, see the manual [Gmb14]. Unlike the Velodyne VLP-16, this LiDAR's emmiters swings from side to side.

8.1.2 Comparison to the VLP-16

The first, most important difference, is that the VLP-16 has significantly more firing planes as well as greater vertical field of view. This makes the LUX4 less effective in terms of scanned points/time ratio. LUX4 has slightly higher frequency of the swings and better angle resolution. The LUX4 also has an option to switch to uneven distribution of the horizontal angle resolution, so called focused scanning. That means it has greater horizontal resolution near the optical axis and it lowers as the beams grow away from it. Detailed description in the manual [Gmb14].

When mounted on a UGV, the VLP-16 has far better perfomance. When it passes some object, it scans it from the side facing the vehicle as well as the LUX4 does. But unlike the LUX4, the VLP-16 scans the side of the object too Afterwards, when the vehicle passed the object, additionally it scans the object's back side. Big advantage of the VLP-16 is that it has wider field of view, both upwards and downwards, therefore it can scan the ground closer

to the vehicle and higher object like trees, lamp posts etc. from much closer distancies to achieve theirs full height to be scanned.

We tried to demonstrate these facts on a static scan with both LiDARs under the same conditions. We did not use a surveying model as the pending issues with RTK might devalue the comparison. A LiDAR was pointed at a window and swinged from a "facing down position" upwards, scanning approximately 70° vertically. In figure 8.3, we can see differencies. Moreover, the VLP-16 also scanned the other side of the room, as it has full horizontal field of view. See figure 8.2.

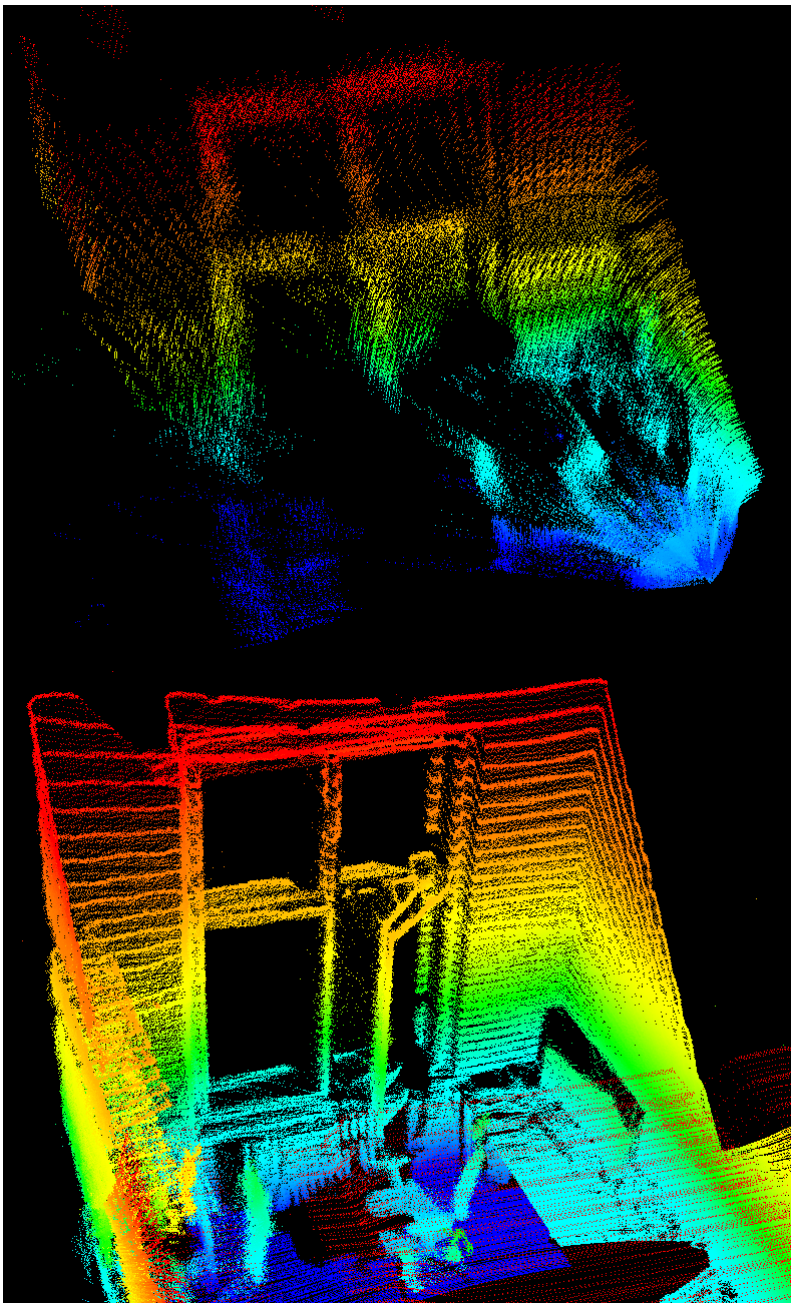


Figure 8.3: LiDARs models comparison

8.2 Photogrammetry

8.2.1 Principle

Photogrammetry is a method to recover depth (distance) information from multiple camera images and create 3D models. It is being used in various scales. It can be used for model construction of a statue, one separate building or some other localy bounded structure in high resolution. On the other hand, it is used to model a terrain or a number of structures or both together. This approach is used for getting an approximate surface structure into a map. A camera can be mounted onto a UAV for this purpose.

8.2.2 Comparison with the LiDAR method

This method has an advantage of adding real textures to the final model. On the other hand, when a LiDAR is integrated into a precise system, it can deliver far more accurate measurements of the scanned area with much less effort (speaking about models of wider areas). With the photogrammetry method, we have to take a lot of photos with big overlaping areas (approx. 80%) and many different angles of the shots to achieve a good result. Also, we have to be aware of using various camera lenses as some of them can round off the images and the correlation algorithms in the photogrammetry software will not work as desired. Finally, a lot of computing power is needed for a final model's construction, again, reducing the "modeled area/time ratio".

The best case with a mapping system is when a LiDAR measures distances (dimensions) of objects and structures and a camera is used to colorize (add texture to) the surfaces. We do not have a UAV mounted with a LiDAR at disposal, so we can only compare a single object model (geometrical stone) created using both mentioned methods. See figure 8.4.

The LiDAR's point cloud was created by merging two separate scans. As we still have some issues pending, the model can be found fuzzy. The photogrammetry point cloud was created using 58 photos (taken in two heights above the ground, all around the stone, by hand). Continuing with the photogrammetry modeling process, we can achieve the model pictured in figure 8.5. The creation of the LiDAR's model took us approximately 10 minutes, while the photogrammetry took around two hours.

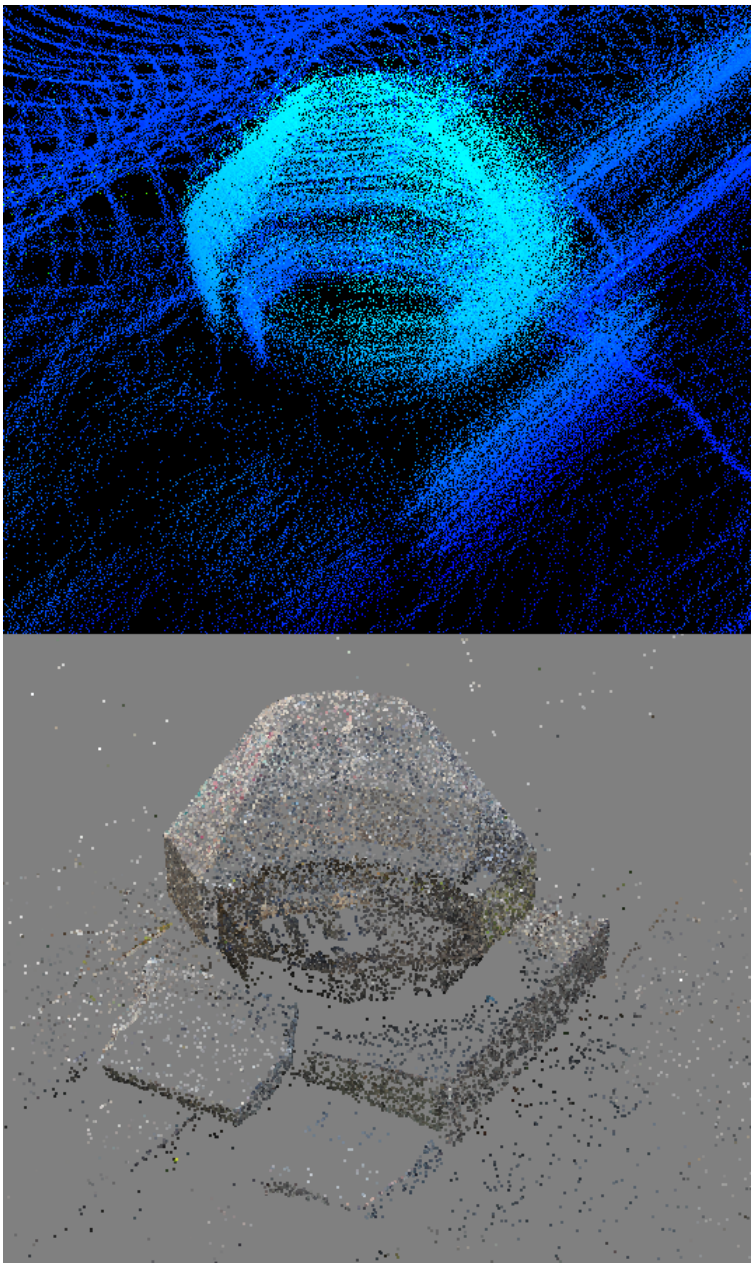


Figure 8.4: Model: LiDAR/photogrammetry

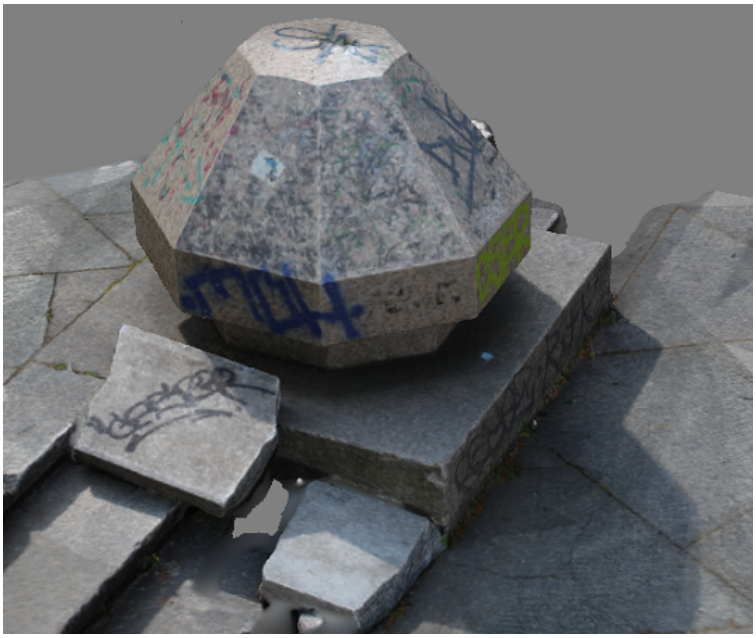


Figure 8.5: Stone model

Chapter 9

Conclusions

We studied the problematics of 3D model construction using the LiDAR technology and chose to integrate an IMU sensor and a GPS module instead of using a SLAM approach. We developed functional communication protocols between the sensors and the computer, and developed algorithms which handle coordinate system's transformations required. After several testing surveys, we were gradually discovering mistakes in our procedures. We tried to remove all imprecisions found. We managed to correct a lot of problems, the pending ones were stated in section 9.2. The comparison with the photogrammetry method is quite inconclusive as we did not reach the desired precision (the best achievable) of the LiDAR's model.

9.1 Project applications

With the restrictive condition that the GPS is only functional outdoors with a clear view at the sky, our system is quite limited in use but still covers a lot of applications. It is being designed mainly for UAVs (or other aerial vehicles).

The main goal, we were aiming for, is a wood volume estimation in forests. With the advantage of a LiDAR, which is capable of scanning through leaves, this technology was the only way to achieve the goal. Aerial scanning system can build a 3D map of a forest, then surface modeling and object recognition algorithms can be applied to highlight the trees and determine theirs volumes.

Our system is usable in any other application that satisfies vehicle's unobstructed view at the sky. It could be: inspection of power lines, management of a land cadastre, 3D modeling of various large-scale constructions (water dams, bridges, castles, ...), field mapping for agricultural machinery automation, landscape examination for construction purposes and so on.

This system in not applicable onto an indoor scanning and a scanning among buildings, where GPS has problems getting unobstructed view on satellites. The SLAM method is much more useful in this type of applications.

9.2 Current problems

After almost a year of development, we are still working on better solutions for the point cloud's construction. We are implementing new hardware connection between a GPS module and the LiDAR to solve the **time synchronization** problem and working on the **coordinate systems misalignment's** solution. In some environments, we have a problem with **ghost points**.

9.3 Suggested future work

We are trying to get the best from the hardware system itself. Our software applications ensure communication with the sensors and agregation of theirs measurements. Final models can be improved and filtered in post-processing to fit requirements of certain application. Also, the data can be used in real-time navigation. The vehicle can navigate itself through the environment, avoid obstacles and reach a desired destination. Moreover, we can write a program which decreases the number of the points in a point cloud while preserving important information, resulting in smaller file sizes. Finally, there could be many possible model processing algorithms which could do e.g. classification of static objects, removal of moving objects, detection of roads (flat areas), detection of some problem (trees growing near power lines, surface of a structure is disrupted), etc.

Appendices

Appendix A

Abbreviations

ARM - Advanced RISC Machine

DSM - Digital surface model

DTM - Digital terrain model

GPS - Global Positioning System

GUI - Graphical User Interface

IMU - Inercial Measurement Unit

LAS - LASer file format

LiDAR - Light Detection And Ranging

LMP - LiDAR's Mounting Point

NED - North, East, Down

RISC - Reduced Instruction Set Computing

RPY - Roll, Pitch, Yaw angles

RTK - Real Time Kinematic

SLAM - Simultaneous Localization And Mapping

VRP - Vehicle's Reference Point

Appendix B

Used technical terms

Azimuth - a synonym for a horizontal angle (yaw)

Data block - contains two firing blocks and has a horizontal angle assigned (VLP-16 data packet)

Firing block - a block of 16 laser firings, one from each plane (VLP-16 data packet)

Firing plane - an area which is being scanned under a single vertical angle

Laser firing - one shot of a laser beam from one emmiter under certain angle, in case of VLP-16, only one laser firing can happen at a time

Appendix C

Bibliography

[Bri08] Lionel Brits, *Euler angles*, ONLINE, January 2008, https://en.wikipedia.org/wiki/File:Eulerangles.svg.

[DWFIB06] Hugh Durrant-Whyte, Fellow, IEEE, and Tim Bailey, *Simultaneous localisation and mapping (slam): Part i the essential algorithms*, September 2006.

[FP10] Tully Foote and Mike Purvis, *Standard units of measure and coordinate conventions*, ONLINE, October 2010, http://www.ros.org/reps/rep-0103.html.

[GMb08] Ibeo Automotive Sensor GMbH, *Reliable in all weathers - the ibeo lux lasescanner for the automotive sector*, ONLINE, March 2008, http://www.abott-mf.com/pdf/ibeo

[Gmb14] Ibeo Automotive Systems GmbH, *Operating manual ibeo lux 2010 laserscanner*, 1.2 ed., 2014.

[Inc15] Velodyne Acoustics Inc., *Vlp-16 user manual and programming guide*, 63-9243 rev. a ed., 2015.

[LiD15] Velodyne LiDAR, *Velodyne_lidar_puck_r_hand_nonenhanced_900*, ONLINE, July 2015, http://velodynelidar.com/images/products/vlp-16/Velodyne_LiDAR_Puck_VLP-16_Images_PNG.zip.

[LLC14] North Coast Media LLC, *Gx4-45_perspective_w*, ONLINE, August 2014, http://gpsworld.com/wp-content/uploads/2014/08/GX4-45_Perspective_W.jpg.

[LOR14] LORD Corporation, *Lord user manual, 3dm-gx4-45*, document 8500-0041 revision d ed., 2014.

[Př16] Jan Předota, *Lidar based obstacle detection and collision avoidance in an outdoor environment*, Bachelors thesis, 2016, Czech Technical University in Prague.

[Swi13] Swift Navigation, *Piksi datasheet*, version 2.3.1 ed., 2013.

[Tů16] Zuzana Tůmová, *Přesná lokalizace malých bezpilotních prostředků s využitím gnss*, Bachelors thesis, 2016, Czech Technical University in Prague.

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

BACHELOR PROJECT ASSIGNMENT

Student: **Tomáš Trafina**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Bachelor Project: **Construction of 3D Point Clouds Using LiDAR**

Guidelines:

1. Study the problematics of 3D model construction using data from LiDARs
2. Identify sensors required for 3D model creation and integrate them on a system level
3. Identify possible sources of imprecision and uncertainty in the measurements
4. Design and implement methods to minimize this imprecision
5. Verify the designed system with Velodyne and Ibeo LUX LiDAR sensors and discuss difference between the measurements.
6. Compare point cloud models with photogrammetric 3D models

Bibliography/Sources:

1. Olson E.: A Passive Solution to the Sensor Synchronization Problem. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.
2. Westra E.: Python Geospatial Development. Packt Publishing Ltd, Birmingham, UK, 2010.
3. Wallace L. at al.: Development of a UAV-LiDAR System with Application to Forest Inventory. Remote Sensing, Vol. 4, Issue 6, Pp. 1519-1543, 2012.

Bachelor Project Supervisor: Ing. Milan Rollo, Ph.D.

Valid until the summer semester 2016/2017

L.S.

prof. Ing. Michael Šebek, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 24, 2016