Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische
Friedrich-Wilhelms-Universität Bonn

Master Thesis

Point Cloud Intensity Compensation and Application in Localization

Author:
Jannis Horn

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
PD. Dr. Lasse Klingbeil

Date: September 26, 2023

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date   Signature

Abstract

Most LiDAR-based odometry systems use only geometric information extracted from point clouds. Modern LiDAR scanners provide further per-point information such as an intensity measurement of the reflected signal. This information could be used to improve localization performance. The measured intensity depends, e.g., on the distance to the hit surface, the rays' incidence angle and the surface reflectivity. This results in differing intensity estimates for the same surface measured from various vantage points. Hence, correct intensity correspondences may be difficult to establish for registration methods, as they require similar intensity.

This thesis investigates the characteristics of the LiDARs' intensity channel and its application for LiDAR-based odometry. This work develops a set of compensation models and a corresponding approach to estimating the parameters. Using range measurements combined with incidence angle and sensor specific compensation visibly improves intensity values, resulting in more even intensity on similar surfaces. To test compensated intensity for localization an intensity residual is added to FastLIO2. First input points are filtered and the intensity compensation is computed. The residuals are generated by estimating local intensity gradients on the map plains used by FastLIO2 for geometric residuals. Tests show that the intensity residuals can improve localization performance, while still retaining real-time performance.

Contents

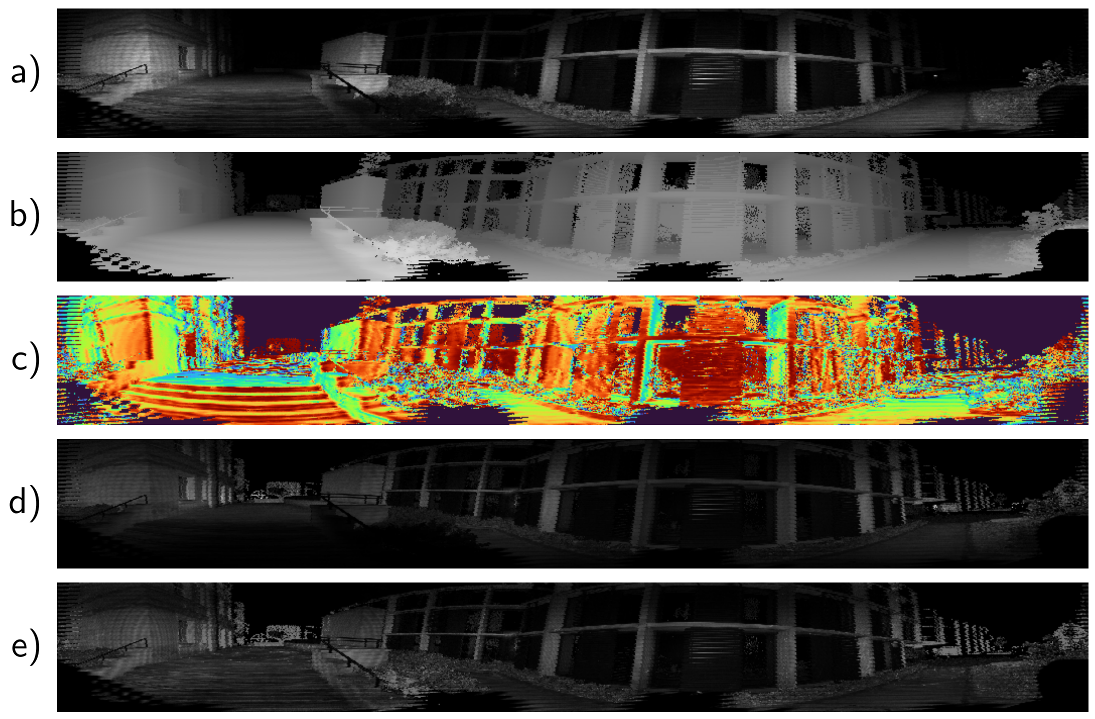

Figure 1.1: a) Raw intensity $I_{\boldsymbol{p}}$, e. g. measured by an Ouster OS0 LiDAR, strongly depends on b) range r and c) incidence angle $\cos(\alpha)$. d) The "reflectivity" channel $\bar{\rho}$ reduces these effects, but remains inconsistent over larger surfaces. e) We estimate a consistent pseudo reflectance $\hat{\rho}$ and extend FAST-LIO2 with pseudo reflectance residuals to improve its accuracy.

1 Introduction

LiDARs ability to accurately measure range led to their broad adaptation in geodesy [Kashani et al. 2015] for land surveying and mapping with airborne (ALS) or terrestrial laser scanner (TLS) on the ground. Autonomous robots are dependent on accurate environment information, for orientation, mapping and localization. For this purpose short range LiDAR sensors are usually employed. These LiDAR scanners provide a wide field of view and high quality range estimation.

LiDAR usually also provides a measure of the return energy: Intensity. This has seen uses e. g. for colorization, land cover classification [Höfle and Pfeifer 2007; Jutzi and Gross 2009; Yan et al. 2012], fault and damage detection after natural disasters [Kashani et al. 2015] or for robot localization in prior maps [Barsan et al. 2020; Hata and Wolf 2014; Levinson and Thrun 2010].

Establishing point correspondence using intensity is difficult, as intensity depends not only on the surface hit, but also on a number of factors such as range to target, incidence angle or sensor intrinsics. To use intensity in localization the points only need to correspond between sensor scans. Accordingly retrieving the real surface reflectance is not necessary.

Using the classification proposed by Kashani et al. [Kashani et al. 2015], intensity correction and normalization should suffice. This includes compensating intensity such that it minimizes non-surface based variations between points in a single scan. Normalization than seeks to ensure that this holds across scans. Luckily for short range LiDAR applications the last step is usually trivial, as atmospheric factors have little effect on short range applications.

Accordingly this work employs an exponential range and angle model proposed by Jutzi and Gross [Jutzi and Gross 2009] for scan geometry compensation and augments this approach by sensor specific compensation modules, such as a short range compensation.

Without the need for real reflectance values, compensation model parameters can be estimated using raw sensor scans. Multiple raw LiDAR scans are fused using preregistered poses. If this fused point cloud contains enough variation in incidence angle, range etc. usable model parameters can be estimated. It might be possible to generate model parameters, using real world data combined with simple guidelines to maximize usable points, without a specialized calibration scene.

To test intensity in localization this work extends the state-of-the-art inertial odometry algorithm FastLIO2 [Xu, Cai, et al. 2022] to not only use geometric information, but also compensated intensity. First a compensation node running in parallel to the odometry estimation, compensates the incoming point clouds.

Using the estimated local planes, utilized by FastLIO for point-to-plane distances, a local intensity gradient on said plane is estimated using found close map point intensity. This gradient can then be used to estimate an intensity residual.

As will be shown usable intensity compensation parameters can be estimated. using non-specialized data, altough with some restrictions. Using this compensated intensity, can improve the localization performance of FastLIO2 and the resulting maps display more consistent intensity.

2 LiDAR Fundamentals

2.1 LiDAR

LiDAR sensors allow to measure the distance between the sensor and the environment. The transmitter sends out a laser using usually an infrared wavelength. The laser is scattered when hitting an object. Parts of the scattered laser are captured by the sensor. This signal is received by a detector installed in the LiDAR sensor, which is build to react to signals of the used laser wavelength. To reduce the effect of ambient lightning the laser band-width is usually chosen to be within wavelengths with little ambient light. Some modern sensors like the Ouster OS0 may also use wave-length with larger ambient footprint, combined with other methods to account for non-laser derived signals.

Two types of LiDAR system are commonly employed: Beam steering LiDAR uses a set of laser emitters and detectors which rotate or are otherwise steered to scan different areas of given scene. This is achieved by either using actually moving functional components or by steering the send and return signal using e. g. mirrors or lenses. Flash LiDAR uses a laser source to illuminate a larger area at once. The resulting backscattered signals are received by detector systems akin to a camera sensor and create an image of an actively illuminated scene.

Modern MOSFET-based LiDAR sensors use a combination of beam steering and flash. These sensors use a single laser source. The laser is projected trough a set of lenses and mirrors. These elements are steered towards the angle to be scanned. There the laser pulse is fanned out according to the sensor design vertical scan angle and illuminates a small area at the same time. Fig. 2.1 shows the described principle.

The reflected laser signal is send back trough a lens aperture. This aperture directs the laser beams towards the correct detector element. In MOSFET-based detectors these detectors are tightly packed on-die, often with small spacial offsets orthogonal to the detector line. This increases packing density and accordingly vertical resolution. Without the offset signal that should be received by one detector might bleed into adjacent detectors and introduce noise into the measurements. These offsets will results in vertical scans of points which to not perfectly align along a vertical line in reality.

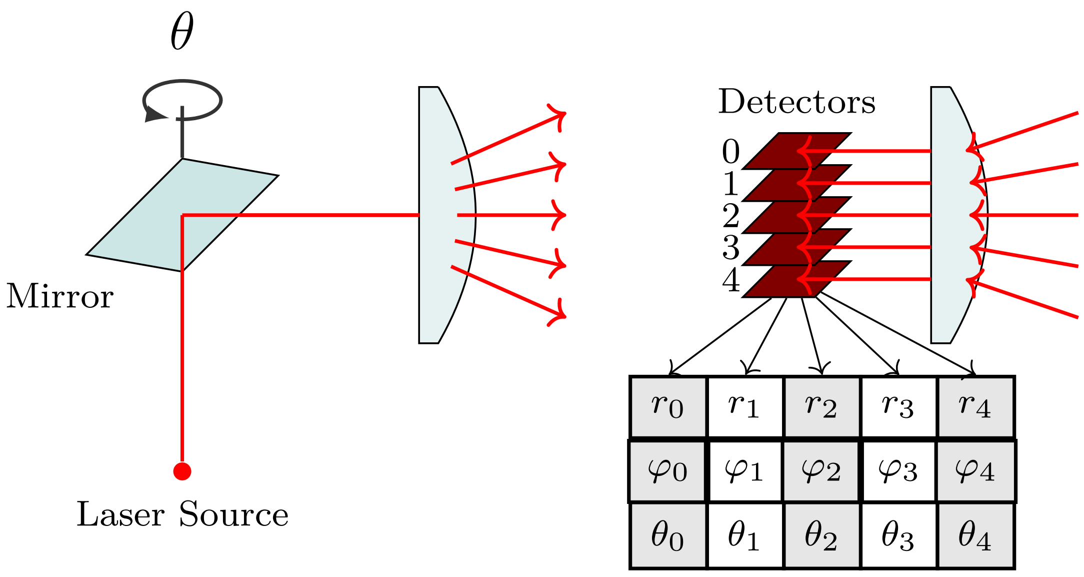
θ
Mirror
Laser Source
Detectors
0
1
2
3
4
r_0 r_1 r_2 r_3 r_4
φ_0 φ_1 φ_2 φ_3 φ_4
θ_0 θ_1 θ_2 θ_3 θ_4

Figure 2.1: Beam-steering flash LiDAR principle. A single laser source is used to illuminate an area in front of the sensor. A set of stacked detectors each generate a single range and intensity measurement, which combine to a single vertical scan line. Horizontal angle θ and verical angle φ are also provided. Note that while θ is the same for a vertical scan, in practice tightly stacked detectors will result in slight deviations in θ for each detector.

Each measurements measures the round trip duration from send and received signal. Accordingly given a laser puls with recorded round-trip distance τ and light speed c:

$$r = \frac{1}{2} c \cdot \tau \tag{2.1}$$

Should τ exceed the scan time Δt no measurement can be taken. How such a case is transformed to data output is sensor dependent. Some sensors report these points as $r = 0$ or other invalid values. Another way may be to not include such measurements at all. Following, the assumption is made that invalid points are included in the datastream and can clearly be identified as invalid.

Given the resulting range measurement r, the corresponding angles (θ, φ) are defined by the read detector and the current orientation of the sender/receiver element. These measurements are transformed to a position $\boldsymbol{p} = (x, y, z)^\intercal \in \mathbb{R}^3$

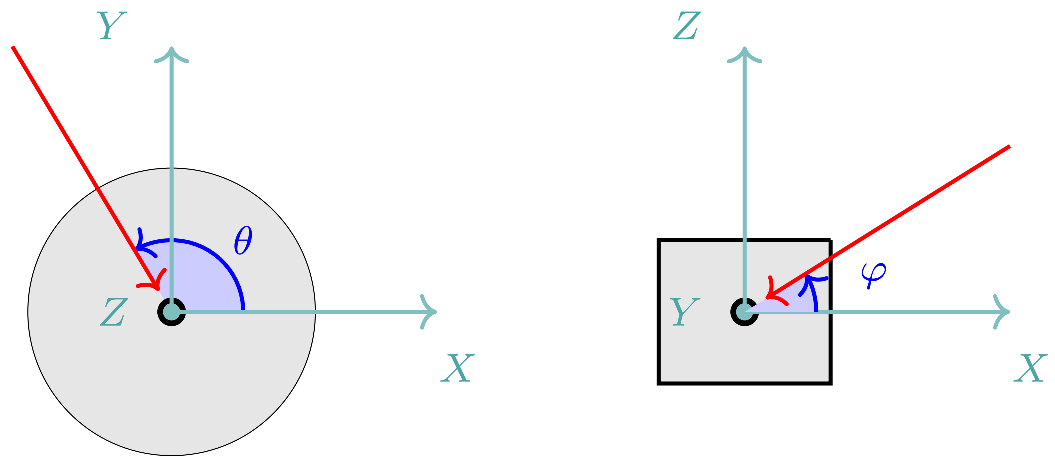

Figure 2.2: The returning laser is recorded at horizontal angle θ and vertical angle φ. The angles are measured in sensor frame. In the idealized model the returning laser hits the sensor directly without redirection. As can be seen in Fig. 2.1 the incoming rays are often redirected.

using the sphere to Cartesian coordinate transformation:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cdot \cos(\theta) \cdot \cos(\varphi) \\ r \cdot \sin(\theta) \cdot \cos(\varphi) \\ r \cdot \sin(\varphi) \end{pmatrix} \tag{2.2}$$

These points are accordingly defined in sensor-frame. The z-axis is the axis of revolution. The position $\theta = 0$ id sensor model dependent. $\varphi = 0$ is usually the x,y-plane. An idealized relationships between (x, y, z), r, θ and φ can be seen in Fig. 2.2.

The actual transformation depends on the sensor construction. Depending on parameters such as lens geometry, aperture geometry, mirror geometry etc. the laser might travel a short distance before being redirected towards its final orientation. These parameters have to be taken into account for retrieving the positions but are highly dependent on the actually used sensor.

2.2 Point Clouds

When further talking about a scan, unless otherwise specified, a full LiDAR scan sweep is described. In short range LiDAR applications this is usually the data generated by a full 360° laser sweep. The range measurements are transformed to positions and assembled in a single point cloud $\mathcal{P}$. Fig. 2.3 shows such full LiDAR scans sampled at different position taken from the Newer College dataset

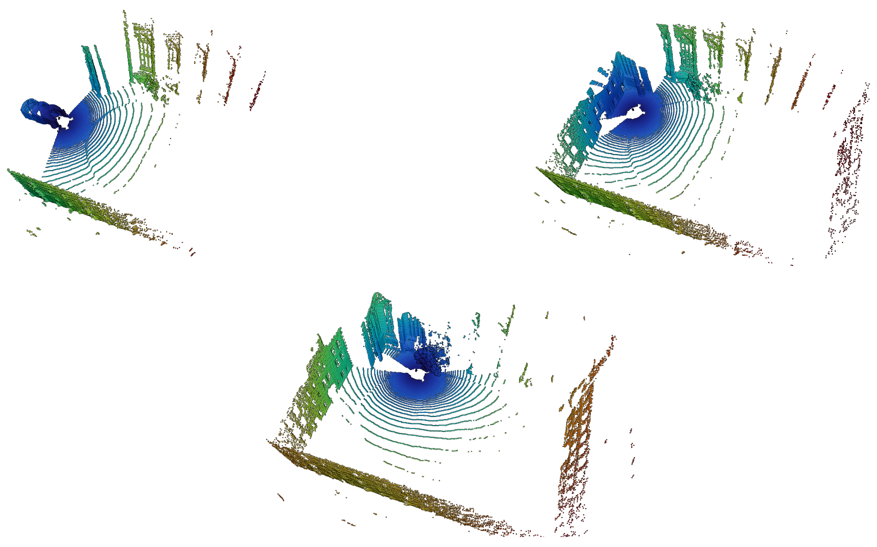

Figure 2.3: Single scan point clouds taken from the Newer College Dataset [L. Zhang, Camurri, and Fallon 2021]. Three point clouds taken from the Quad-Easy dataset. The clouds are colored according to point distance to the sensor with blue being close. Points with invalid range measure are not shown. Note that the blind spot at the sensors back is due to the cable mount.

[L. Zhang, Camurri, and Fallon 2021].

A multi-beam LiDAR system records h measurements simultaneously. Each detector has a fixed vertical angle φ while the horizontal angle θ varies equidistantly w times throughout a full revolution. The horizontal slices are measured one at a time. Hence an assembled point cloud $\mathcal{P}$ consists of slices recorded at slightly different times. During sensor movements the measurement origin $\boldsymbol{o}$ may change and distorts the point cloud $\mathcal{P}$ if the motion is not compensated. This may be necessary for fast moving applications.

Each point cloud row is called a ring $\nu \in [0, h-1]$. As points belonging to the same ring, are recorded using the same detector they share the same internal signal path for sending and returning signals. This parameter is recorded for each point. It is eiter provided by the sensor directly or can be inferred from φ or, if the scan line is complete and ordered, by its iterator.

All measurements may be ordered as a $w \times h$ image. Using the assumption made for invalid points the, resulting image representation is dense. Hence the ring and horizontal position of every point can be directly retrieved from its iterator i in

$\mathcal{P}$. Assuming the point cloud is ordered in row-major order:

$$\nu_i = \lfloor \frac{i}{w} \rfloor, \tag{2.3}$$

$$w_i = i \bmod w. \tag{2.4}$$

LiDAR Point

A point cloud p with vertical resolution h and horizontal resolution w consists of $w \times h$ points $\mathbf{p}$. Each point consists of the sensor provided ring ν, range r, intensity I and for certain sensors reflectivity $\bar{\rho}$. These are augmented by the point position parameters that can be computed using just $\mathbf{p}$. The local point normal $\boldsymbol{n}$ is the normal of the local plane. This can be estimated using the local point neighborhood. Using $\boldsymbol{n}$ the local incidence angle α can also be computed, see also Sec. 4.1.

Putting these together, a point is defined as:

$$\mathbf{p} = [\nu, r, I_{\boldsymbol{p}}, \bar{\rho}, \boldsymbol{p}^\intercal, \boldsymbol{n}^\intercal, \alpha]^\intercal \,, \tag{2.5}$$

2.3 LiDAR Intensity

Most LiDAR sensors provide not only positional data, but also a measure of received energy. A detector receiving a back scattered laser returns this energy as LiDAR intensity I for each point. The hit surface will absorb some of the received energy and reflect the remainder. This ratio is a near constant surface characteristic and can be used to distinguish between different surface types.

The amount of reflected energy is the surfance reflectance ρ. Reflectance changes depending on the reflected wavelength λ and incidence angle α. As LiDAR uses direct illumination with small bandwith, λ is assumed to be a single value depending on the sensor model.

The surface gloss η describes the amount of specular reflection. Gloss describes the ratio of diffuse to specular reflection. High gloss results in high directional reflection, while high diffuse reflection refracts the light more evenly independent of α. Gloss cannot be estimated by single point observations. Following lambertian reflectance is assumed for reflectance measurements.

Surface reflectance ρ is the amount of energy reflected when the surface is hit by electromagnetic waves and may hence allow to distinguish between different surfaces. This can be of use for feature extraction, classification and odometry. To

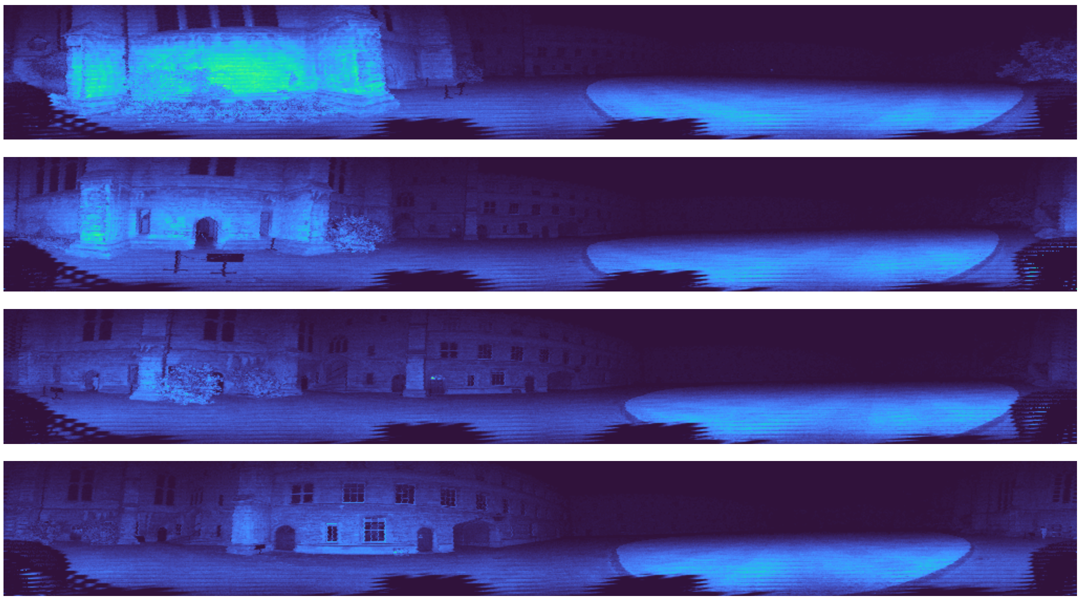

Figure 2.4: Multiple $2d$ projections taken from the Newer College Quad-Easy dataset. Images are colored by the recorded intensity value clamped to range $[0, 0.35 \cdot I_{max}]$. The scans are taken 40 LiDAR frames apart. The image center is the forward direction for sensor movement.

do so ρ or another measure identifying the hit surface should be extracted from intensity I.

2.3.1 Measurements Parameters

Kashani et al. [Kashani et al. 2015] list four categories of factors influencing LiDAR intensity: Target Surface Characteristics ρ and η, Acquisition Geometry r and α, Instrumental Effects η_{sys} and Environmental Effects η_{atm}. Fig. 2.5 shows an overview where these parameters interact with intensity acquisition.

Target Surface Characteristics

The target surface influences intensity when the laser is reflected from the hit surface depending on the surface reflectance ρ and gloss η.

Surfaces with high gloss can result in very high return energy. Intensity measurements on close objects may results in very high signals. High gloss objects can be detected over longer ranges compared to lower gloss surfaces. The high reflectance may lead to return signals not only hitting the designated detector, but also adjacent detectors leading to a bloom effect. This may override the correct laser signal or lead to loss of return signal. Examples of this can be seen in

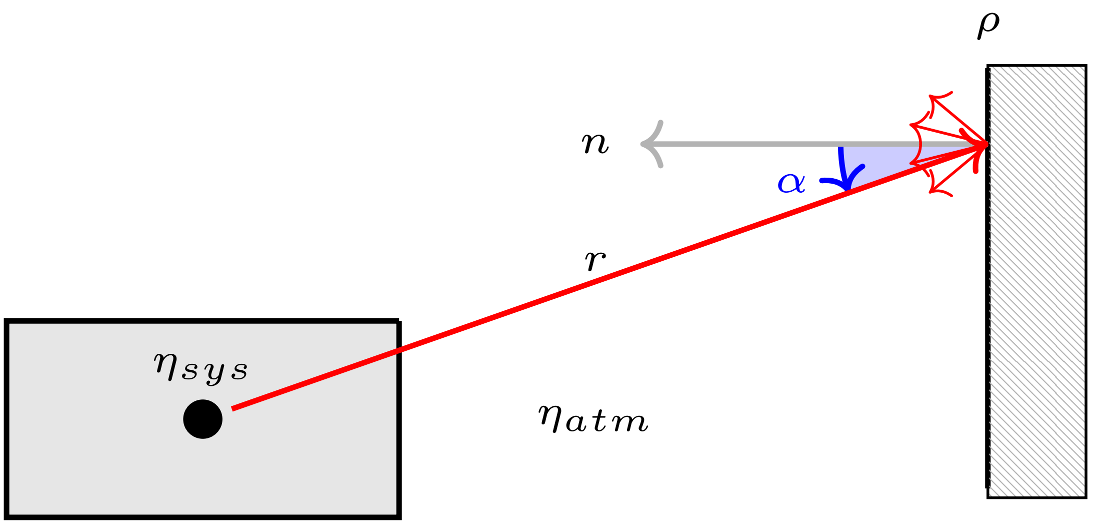

Figure 2.5: Measured intensity depends on four distinguishable parameters: η_{sys} the sensor intrinsic parameters such as laser energy or aperture construction, η_{atm} atmospheric effect e. g. temperature or air pressure, surface reflectance ρ and gloss η and acquisition geometry range r and incidence angle α.

Fig. 2.6.

Acquisition Geometry

Laser energy is lost during travel. This is e. g. due to atmospheric absorption or scattering. Another important factor is beam divergence. During travel the laser expands and increases the area under the laser. This reduces energy density. Accordingly the measured intensity I reduces with increasing range r.

The second geometric parameter is incidence angle α. Assuming Lambertian reflectance the energy reflected by a hit surface decreases with growing incidence angle. High gloss η may further amplify this effect, as more energy undergoes specular instead of diffuse reflectance.

Instrumental Effects

These encompass all effects derived from the sensor setup. The used laser emitter influences the emitted initial energy as well as wave length and hence the measured absorption spectrum. The detector structure decides how much energy can be received, as well as how this measure is scaled and represented. Depending on sensitivity this might also effect noise in measurements.

Internal lenses and mirrors both further influence send and received energy.

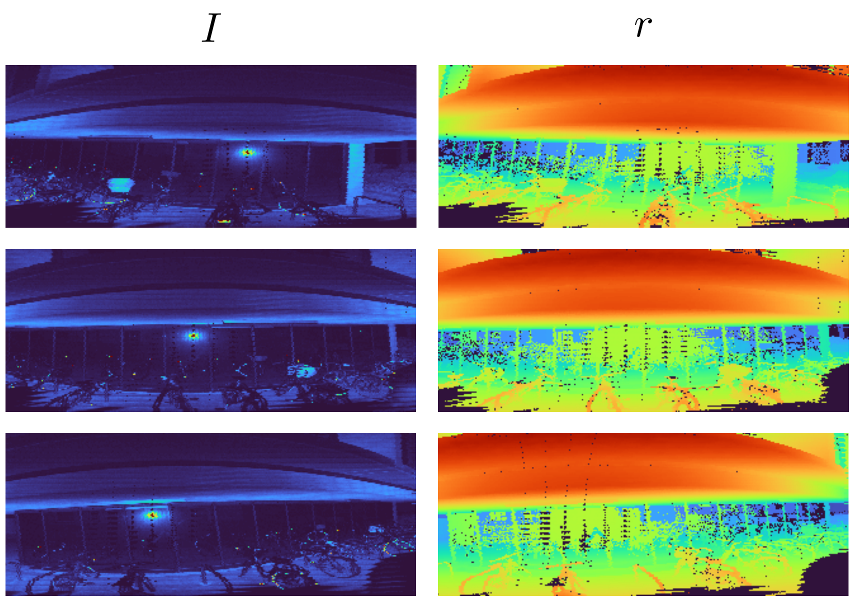

Figure 2.6: Cropped $2d$ projections of point clouds taken from Newer College Math-Medium. Left: Point intensity clamped to $[0, 0.35 * I_{max}]$. Right: Measured range per point from colored red/yellow for short range to blue for far ranges. Scanning non-lambertian surfaces like glass results in very different results depending on incidence angle α. For small α the glass returns high energy reflections. Points surrounding these areas are often lost.

Possible effects are e. g. vignetting introduced by lense structures or loss of energy for imperfect transmitting components.

Sensor specifications also decide how intensity data is encoded. Depending on model this is usually either as integer (8- or 16-bit) or floating-point number. All these factors are combined in η_{sys}. While some components such as output format are usually provided by the manufacturer, other factors such as internal transmittance or laser specifications are often not publicly available.

Environmental Effect

The transmitting medium (usually atmospheric air) can introduce further effects. This consists of (atmospheric) transmittance η_{atm} influenced among other things by humidity, temperature and pressure. Atmospheric effects may increase range based effects and add further effects based on range. This factor is primarily of interest for long range laser measurements and according to Kashani et al. 2015 can be assumed to be constant for short range $r < 50m$ applications.

2.3.2 LiDAR Range Equation

These factors are formalized by Jelalian [Jelalian 1992]. The proposed model is the laser radar range equation or LiDAR range equation, which desribes the relationship between the aforementioned factors:

$$P_r = \frac{P_t D_r^2 \eta_{atm} \eta_{sys} \sigma}{4\pi r^4 \beta_t^2} \tag{2.6}$$

P_r is the returned energy. P_t is the send energy and D_r the detector aperture diameter. While these factors are sensor intrinsics, their influences are universal for all setups and therefore are listed separately. β_t is the laser transmit beamwidth measured in radians. While the laser beam divergence increases over time the beamwidth angle does not.

The target cross-section σ describes the interaction between target and laser energy. This combines the surface reflectance ρ with the local scattering solid angle Ω. Ω is usually position dependent and may not be consistent throughout the surface. This also encompasses the effects of gloss η. Higher η results from flatter scattering solid angles. A_t is the target area hit by the laser beam.

$$\sigma = \frac{4\pi}{\Omega} \rho A_t \tag{2.7}$$

Ω or A_t cannot simply be retrieved just using LiDAR scanners. To get a reliable value, further sensor system or surface classifications are necessary. Another way would be exploratory actions with LiDAR sensor to gather multiple scans of the same surface.

Simplified LiDAR Equation

To address these parameter, a set of assumptions are made. The first assumption used e. g. by Höffle and Pfeifer [Höfle and Pfeifer 2007] assumes $\Omega = 2\pi$. This is equivalent to the surface refracting incoming beams into a half-sphere. The second assumption made is that the laser completely hits a single target. Accordingly the target area A_t can be computed only using beamwidth β_t and range r. The last assumption made is lambertian-reflectance which adds a factor of cos α [Jutzi and Stilla 2006].

$$A_t = \frac{\pi r^2 \beta_t^2}{4} \tag{2.8}$$

$$\sigma = r^2 \beta_t^2 \rho \cos(\alpha) \tag{2.9}$$

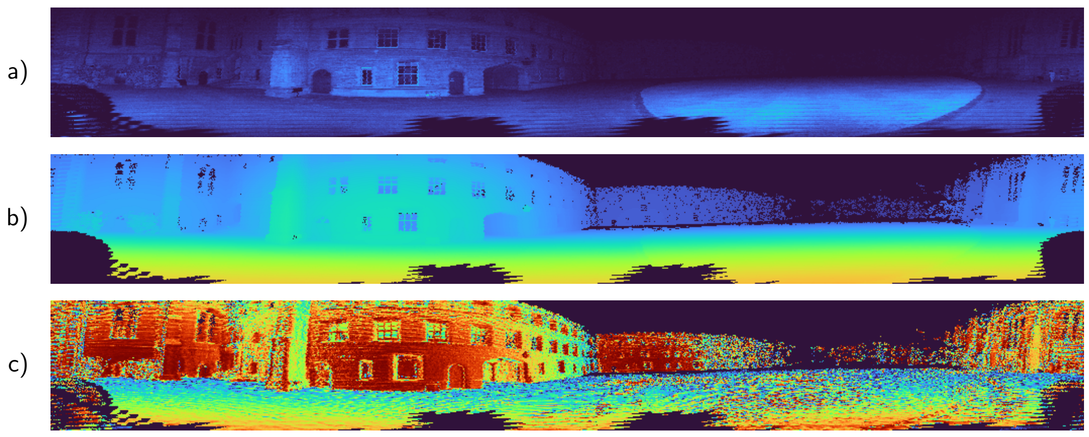

Figure 2.7: $2d$ projections of a single point cloud taken from Newer College Quad-Easy. a) Point intensity I clamped to $[0, 0.35 * I_{max}]$. b) Measured range r per point from colored red/yellow for short range to blue for far ranges. c) cos α per point. α is estimated using local normal estimations. As can be seen I quickly decreases with increasing r.

The resulting simplified LiDAR Equation now looks as follows:

$$P_r = \frac{P_t D_r^2 \eta_{atm} \eta_{sys} \rho \cos(\alpha)}{4r^2} \tag{2.10}$$

To identify surfaces, reflectance ρ is of interest. Further P_t, D_r and η_{sys} are sensor model specific and can assumed to be constant for scans using the same sensor. Rearranging the Eq. 2.10 yields:

$$\rho = \frac{P_r}{P_t D_r^2 \eta_{sys}} \frac{4r^2}{\eta_{atm} \cos(\alpha)} \tag{2.11}$$

$$\rho \propto \frac{r^2}{\eta_{atm} \cos(\alpha)} \tag{2.12}$$

While this theoretical formulation describes the relationship between the measured energy and dependent parameters, it can be challenging to get good estimations of these parameters. Both instrumental effects η_{sys} and environmental effects η_{atm} can also introduce further non-linearity into the model. This may be caused by non-linear receiver response or internal laser signal transmission. These parameters may also be indirectly depending on geometric effects, as the change in energy changes how the sensor system reacts besides the expected sensor response.

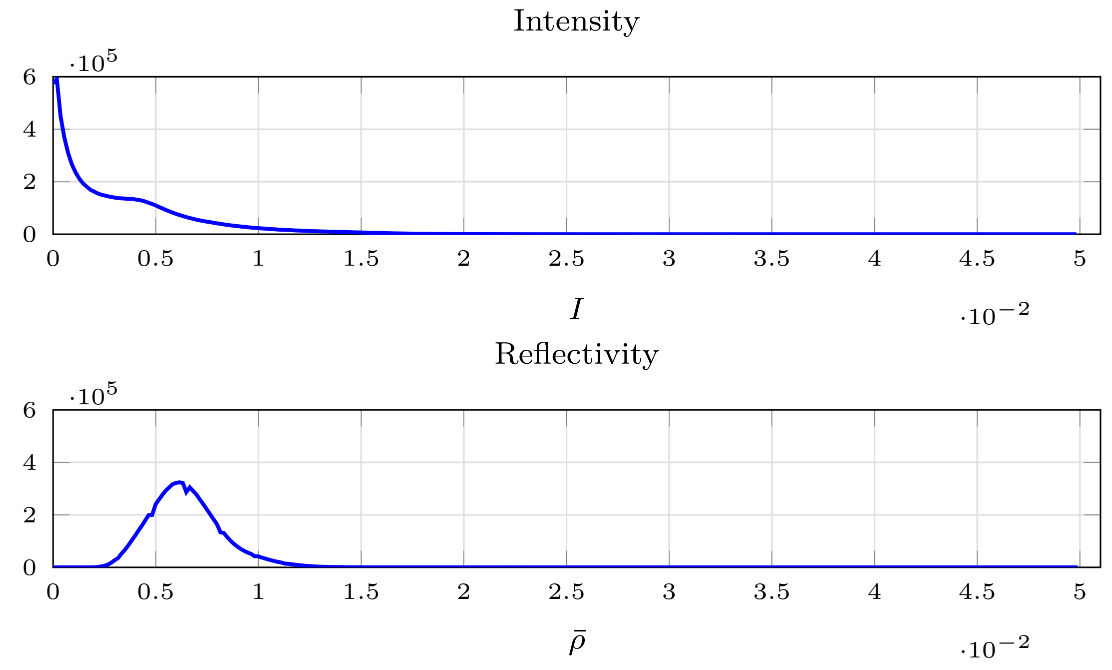

Figure 2.8: Further information channels recorded with an Ouster OS0. Top shows the intensity and Bottom the sensor provided reflectivity. Measurements where taken from form the Newer College Quad-Easy dataset [L. Zhang, Camurri, and Fallon 2021].

2.3.3 Reflectivity

Some modern LiDAR scanners provide more than just distance/3D point information and intensity I. Many modern LiDAR sensors provide precompensated intensity. The Ouster OS0 for example provides the reflectivity channel $\bar{\rho}$.

This compensation usually apply a custom compensation function based on range r. The compensation also depends on correct calibration, as the exact magnitude usually depends on sensor construction. Intensity measurements are often clumped towards the low values. To increase intensity resolution, precompensation can include rescaling of the compensated intensity. Fig. 2.8 shows the initial intensity distribution and the applied precompensation.

The reflectivity channel provided by the Ouster OS0 sensor compensates the intensity by range and then rescales this reflecivity by a linear function for low values and a log function for higher values. Cutoff for older models was at $I = 100$. Fig. 2.9 shows $\frac{\bar{\rho}}{I}$ against r. The r compensation done, can clearly be seen starting at 4m.

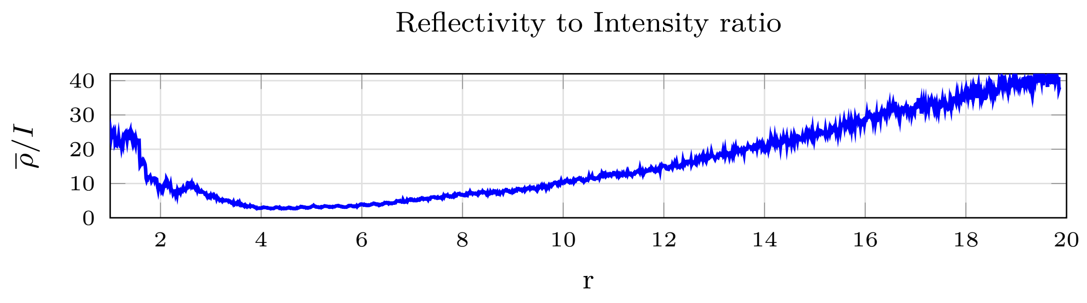

Figure 2.9: Comparison of reflectivity to intensity measured using an Ouster OS0. Points are ordered according to their range measurement. The distance compensation applied can clearly be seen.

3 Related Work

Intensity Compensation

A lot of work on intensity correction, normalization and calibration has been done for airborne LiDAR systems, as intensity can be used to classify ground areas.

Höffle and Pfeifer [Höfle and Pfeifer 2007] utilizes both a data driven and a theoretical model to compensate intensity data. The data driven approach tries to estimate a set of parameters by optimizing multiple models. Flat ground data from multiple flights are combined and registered against each other to create point pairs. The theoretical model uses Eq. 2.10.

Jutzi and Gross [Jutzi and Gross 2009] combine these approaches. A parameterized model based on Eq. 2.10 is combined with a Phong reflection model [Phong 1998] to account for non-lambertian surfaces. Data of rooftops and other homogeneous areas with varying incidence angle are recorded and registered to optimize the model parameters.

Yan et al. [Yan et al. 2012] again use a physics based model using the Eq. 2.6. To estimate parameters for geometric correction, overlapping scan strips are used. To do so scan poses are compensated for airplane motion. This is combined with a comprehensive atmospheric model. The resulting calibrated intensity is used for surface classification.

Lehner and Briese [Lehner and Briese 2010] measure ground reflectance using spectrometers on pre-registered objects. This is combined with airborne measurements of these same areas. Atmospheric factors are again measured by control station close to the scanning area. After compensation for athmospheric transmittance, the system and geometric parameters can be estimated by comparing the scanned data to the reference.

Levinson and Thrum [Levinson and Thrun 2010] calibrate short range LiDAR systems for use in autonomous driving. Intensity data is collected in a map, which is subdivided into cells. A parameterized model is computed using detector ID, current sensor angle and range. The resulting model is a calibration matrix of size $h \times w$. For each cell the intensity is assumed to be the expectation over all measurements the cell contains. The difference between expectation and measurement is used to optimize the compensation matrix.

Steder et al. [Steder et al. 2015] estimate compensation parameters for short range LiDAR systems. A set of pre-registered point clouds is used. The resulting map is discretized into map cells. Again the target intensity for each cell is assumed to be the mean over all measurements in given cell. The points are combined in a Bayes Network. The network alternately optimizes the map and the compensation parameters.

Intensity for Localization

The calibration approach by Levison and Thrune [Levinson and Thrun 2010] is used for the purpose of road markings detection by Hata and Wolf [Hata and Wolf 2014]. The calibrated intensity is used to find road markings using estimate and variance thresholding over all points inside curb curvesd. The localized road markings are then used to localize a car driving along a road.

Barsan et al. [Barsan et al. 2020] use deep neural networks to register local intensity maps inside prerecorded scenes. The network is trained to create intensity embeddings of local scans and scans taken from the map. The cross correlation between embeddings with probability estimation models for similiarity between estimated pose in LiDAR, GPS and dynamic model is used to estimate the agents position.

Wang et al. [H. Wang, C. Wang, and Xie 2021] propose a SLAM algorithm utilizing compensated intensity. The pre-compensated intensity channel, provided by a LiDAR sensor, is combined with further incidence angle compensation. Features in point clouds are extracted based on combining local geometric and intensity distribution. Map matching is done by optimizing geometric and intensity residuals.

4 LiDAR Intensity Compensation

As discussed in the previous Sec. 2.3 there are multiple different factors influencing the measured intensity. Depending on the application, the intensity is used for, not all of these need to be corrected for. For applications running on specific sensor systems it is often sufficient to only compensate for non-instrumental effects. The resulting compensated intensity will therefor not be the actual object reflectance ρ but the *relative-* or *pseudo-reflectance* $\hat{\rho}$.

4.1 Incidence Angle computation

The laser incidence angle α is a parameter, the theoretical model states is influential regarding laser intensity. As it is not provided by the sensor itself, it has to be computed. To do so the laser incidence angle α the orientation of the surface hit by the laser is necessary. This information is given by the plane normal $\boldsymbol{n}$. To compute a plane normal at least 3 non co-linear points are necessary. Quality of the normal increases with the number of points along the same surface used.

Point-to-surface association is unknown just from the LiDAR scan. Therefore the assumption is made that for given point, surrounding points, that are spatially close, belong to the same surface. This assumption holds for scenes in which most surfaces are wider than $n \cdot w$ and $m \cdot h$. Small objects compared to scanning resolution often only provide a handful of responses. For such objects normal computation cannot be reliably done. The resulting estimation can be very noisy. While this approach only generates approximation of surface normals and tends to smooth edges between surfaces, it tends to work sufficiently well in practice. Results using this approach can be seen in Fig. 4.1.

4.1.1 kNN Computation

When only using a single point cloud ν_i and w_i can be used to approximate the set of closest points. The assumption is made that, for given point most close points are also close wrt. ν_i and w_i. As the input cloud is assumed to be dense and ordered Eq. 2.3 and Eq. 2.4 can be used to quickly retrieve point iterators. Given point cloud $\mathcal{P}$ with height h and width w:

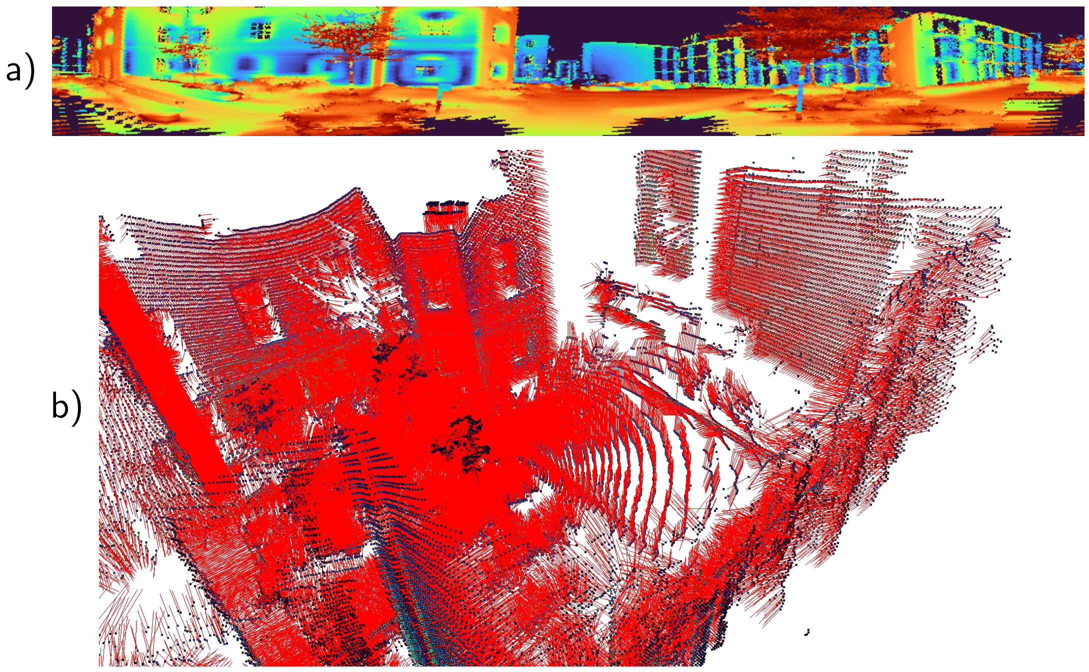

Figure 4.1: Per point incidence angle estimation. The angles are retrieved using one point cloud together with the KD Tree knn estimation. These angles are shown as 2d projection in a). The smoothing effects along the edges are the result of using only nearest-neighbors. The estimated plane normals are shown in b).

```
procedure computeKNNsOrdered($\mathcal{P}$, $d_{knn}$, $h_{diff}$, $w_{diff}$)
    $sqddist_{knn} \leftarrow d_{knn}^2$
    $list_{knns} \leftarrow []$
    for all $\mathbf{p}_i \in \mathcal{P}$ do
        $h_{min} \leftarrow \max(0, \nu_i - h_{diff})$, $h_{min} - 1 \leftarrow \min(h, \nu_i + h_{diff})$
        $w_{min} \leftarrow \max(0, \nu_{i,w} - w_{diff})$, $w_{min} - 1 \leftarrow \min(w, \nu_{i,w} + w_{diff})$
        $list_{\mathbf{p}_i} \leftarrow []$
        for $h_k \leftarrow h_{min}$ to $h_{max}$ do
            for $w_k \leftarrow w_{min}$ to $w_{max}$ do
                $j \leftarrow \text{toIndex}(h_{it}, w_{it})$
                $\mathbf{p}_j \leftarrow pc[j]$
                if $\text{sqrdNorm}(\boldsymbol{p}_i - \boldsymbol{p}_j) \leq sqddist_{knn}$ then
                    $list_{\mathbf{p}_i}.\text{insert}(j)$
                end if
            end for
        end for
        $list_{knns}.\text{insert}(list_{\mathbf{p}_i})$
    end for
    return $list_{knns}$
end procedure
```

This algorithm generates an unsorted list of all neighboring points which are closer than given threshold. This is done by only searching a small frame of the point cloud. The frame is centered around the sample point and is limited by h_{diff} and w_{diff}. Therefor this list is only an approximation which converges towards the correct knn solution with growing frame size. The resulting neighborhood should generate sufficient points to run the chosen algorithm for normal computation.

When computing the k-nearest-neighbors on combined point clouds this approach does not work. For these point clouds a more robust algorithm is necessary. KD-Trees [Bentley 1975] can be used to quickly look up the k-nearest-neighbors for given point.

The algorithm used to compute the KD Tree for given point cloud is nanoflann [Blanco and Rai 2014] based on the original flann [Muja and Lowe 2009] implementation. The c++ library uses static polymorphism as well as some other optimization to allow somewhat efficient KD Tree build and lookup time. This still is much slower than the ordered KNN computation, but results in more accurate normal estimations.

4.1.2 Normal Computation

Given point $\mathbf{p}$ and the k-nearest neighbors $\{\mathbf{p}_1 = \mathbf{p}_i, \mathbf{p}_2, \dots, \mathbf{p}_k\}$ the local normal $\boldsymbol{n}$ can be estimated using the method described by Rusu [Rusu 2009].

A plane normal is orthogonal to the plane. This is equivalent to fitting a plane tangent to the local surface. This can be defined as a least-square fitting problem, with plane anchor $\boldsymbol{q}$ and error function d:

$$\boldsymbol{q}_i = \frac{1}{k}\sum_{j=1}^{k}\boldsymbol{p}_j, \tag{4.1}$$

$$d_i = \sum_{j=1}^{k}(\boldsymbol{p}_j - \boldsymbol{q}_i)\cdot\boldsymbol{n}. \tag{4.2}$$

The least square problem can be solved by principal component analysis over the point positions. This is equivalent to computing the eigenvalues and eigenvectors of the local covariance matrix $\boldsymbol{C} \in \mathbb{R}^{3\times 3}$. As $\boldsymbol{C}$ is symmetric and real the eigenvalues can be extracted by eigenvalue-decomposition:

$$\boldsymbol{C} = \frac{1}{k}\sum_{j=1}^{k}(\boldsymbol{p}_j - \boldsymbol{q}_i), (\boldsymbol{p}_j - \boldsymbol{q}_i)^\intercal \tag{4.3}$$

$$\boldsymbol{C} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\intercal. \tag{4.4}$$

The eigenvector associated to the smallest eigenvector is an approximation of the plane normal $\boldsymbol{n}$.

The direction of the plane normal computed using PCA is ambiguous. It can either point away from the observer or towards it. For incidence angle computation the plane normal should always point from plane into the scene. Therefore´ for all computed normals, should the dot product of normal and distance vector from observer to point be negative, the sign of the normal is inverted.

4.1.3 Incidence Angle Computation

For all points for which the plane normal can be estimated the laser incidence angle can now be computed. The incidence angle is the angle between incoming laser and the plane normal.

A generalized function to compute the angle between two 3D vectors can be given by following formula. Let $x \in \mathbb{R}^3$ and $y \in \mathbb{R}^3$:

$$\alpha = \operatorname{atan2}(||x \times y||, x \cdot y) \tag{4.5}$$

This formulation is reasonably computationally efficient and numerically stable. Reflection should not occur on angles $\alpha > \frac{\pi}{2}$ as such a surface could not be hit. The plane normals are only estimations and might result in surfaces estimates that would be impossible to be hit. The resulting angle $\alpha \in [-\pi, \pi]$ is therefor filtered before further computation.

4.2 Intensity Compensation

4.2.1 Compensation Stages

Kashani et al. [Kashani et al. 2015] proposes three stages of active intensity compensation: correction, normalization and calibration.

Intensity Correction

The goal of intensity correction is to reduce intensity variance in single point clouds. This can be achieved by compensation for some of the parameters such as r and α. This can be done by just comparing values scanned by the same scanner setup. Usually rigorous compensation for η_{sys} is not performed, as it is difficult to estimate system biases just using system outputs. Accordingly intensity correction is not employed to retrieve real reflectance ρ but pseudo-reflectance $\hat{\rho}$.

Intensity Normalization

Intensity normalization is used to reduce inter scan variation in intensity. Methods usually scale contrast and brightness such that overlapping areas share similar values. This is usually necessary if external conditions between scans change e. g. atmospheric transmittance η_{atm}. In short range application this step is of lesser interest.

Intensity Calibration

Calibration is used when the real reflectance ρ should be retrieved from intensity. This includes estimating η_{sys}. To do so, necessitates measuring target reflectance using external instruments. The resulting compensation can then be used to retrieve ρ and allows intensity comparisons between sensors and scan setups. During runtime this is usually augmented with the correction or normalization to compensate for runtime dependent parameters such as r.

4.3 LiDAR Intensity Compensation Models

Full intensity calibration for localization is only necessary, if comparison to preregistered intensity maps is of interest, or if multiple different LiDAR sensors are employed simultaneously. For the purpose of short range LiDAR scans normalization is not usually needed, as η_{atm} can be assumed to be near constant for ranges $r < 50m$. Other parameters do not change contrast or global brightness between scans in the same scan run.

Hence models for intensity correction are applied. The goal of intensity corrected as described in section 4.2.1 is to decrese intensity variation. This is achieved by modifiying intensity values such, that values generated from similar reflective surfaces generate similar intensity values. The relationship between intensity and reflectance is described in Eq. 2.6 and in its simplified form in Eq. 2.10.

Given point position $\boldsymbol{p}$ and model $g(\cdot;\phi)$ parameterized by ϕ, a correction model should return the same pseudo-reflectance $\hat{\rho}$ for point $\mathbf{p}_i$ at $\boldsymbol{p}$ independent of range r_i and incidence angle α. Formally given points $\mathbf{p}_i$ and $\mathbf{p}_j$

$$\boldsymbol{p}_i = \boldsymbol{p}_j \implies g(\mathbf{p}_i;\phi) = g(\mathbf{p}_j;\phi), \tag{4.6}$$

holds for perfect models.

To utilize compensated LiDAR intensity for the purpose of localization, points with similar reflectance should have similar compensated intensity compared to points in other scans taken of the same environment. FastLIO works without the use of prerecorded maps. Therefore full Intensity Calibration is not necessary.

4.3.1 Geometric Compensation

Given point $\mathbf{p}_i$ with intensity I_i, the geometric distance to the sensor r_i is available. If the local plane normal can be computed, the local incidence angle α_i is also available. According to Eq. 2.10 a simple geometric compensation c_{geo} looks like this:

$$g(\mathbf{p};\varnothing) = I_i \frac{r_i^2}{\cos(\alpha_i)}. \tag{4.7}$$

This equation is not defined for $\cos(\alpha) = 0$. A reflection for $\alpha \geq \frac{\pi}{2}$ should not occur, but inaccuracy in the sensor and normal computation may generate larger α. As angles $\alpha \to \frac{\pi}{2}$ $\frac{1}{\cos\alpha}$ grows fast and is more susceptible to noise. Hence points $\mathbf{p}$ with $\alpha > \alpha_{\max}$ are not compensated. In practice $\alpha_{\max} = 1.5$ is used.

As can be seen in Fig. 4.2 the compensate unevenly. While wall areas are

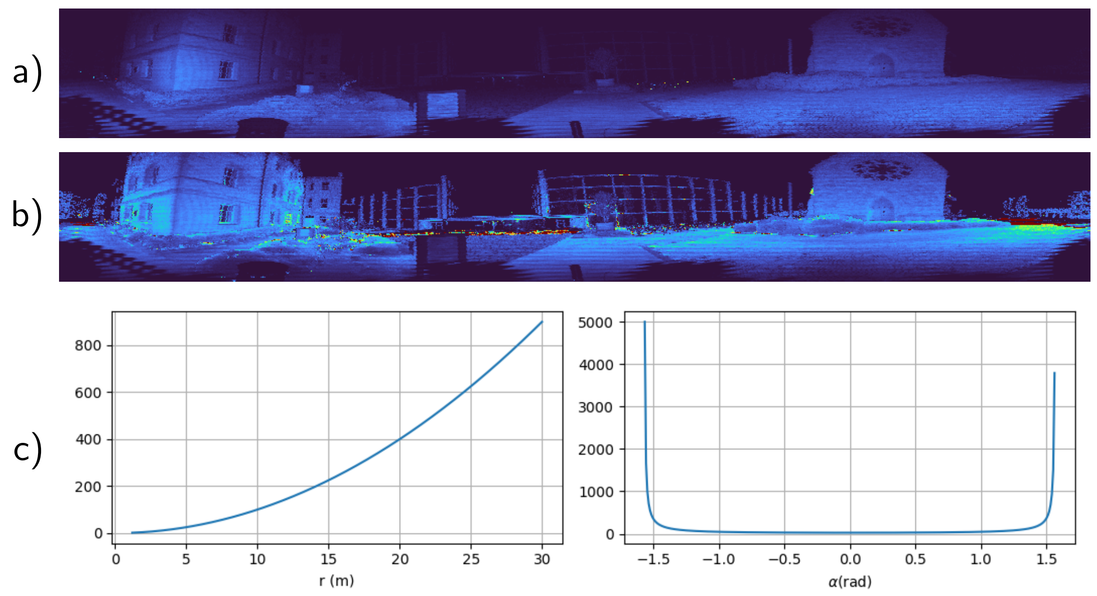

Figure 4.2: a) Intensity clamped to $[0, 0.3]$ and b) compensation intensity clamped to $[0, 5e-5 \cdot I_{max}]$ using the geometric model. While range seems to be compensated for, small noise in the angle computation creates disproportionate compensation in places. While the left walls are somewhat even, most ground areas are overcompensated c) shows the geometric compensation function plotted over range (left) and angle (right).

undercompensated, floor areas in part are overcompensated. Also the steep slope resulting from $\frac{1}{\cos\alpha}$ leads to very noisy areas, where angle estimation is noisy. These problems seem to be primarily due to an overestimation of the influence of $\cos(\alpha)$.

4.3.2 Weighted Range And Angle

To improve on Eq. 4.7 the angle component has to be scaled compared to the range component. The LiDAR Equation does not give insight into this relationship. Eq. 2.10 suggests a multiplicative relationship. Therefor given parameter $m \in \mathbb{R}$ the most basic model is:

$$g(\mathbf{p}; m) = I \frac{r^2}{1 + m \cdot \cos(\alpha)}. \tag{4.8}$$

The parameter m can be optimized for using the framework described in Sec. 4.4. As can be seen in figure 4.3 while ground areas are more even and most noise artifacts are removed. Areas along walls are still uneven. The assumptions made for equation 2.10 may not hold completely or the geometric parameters might

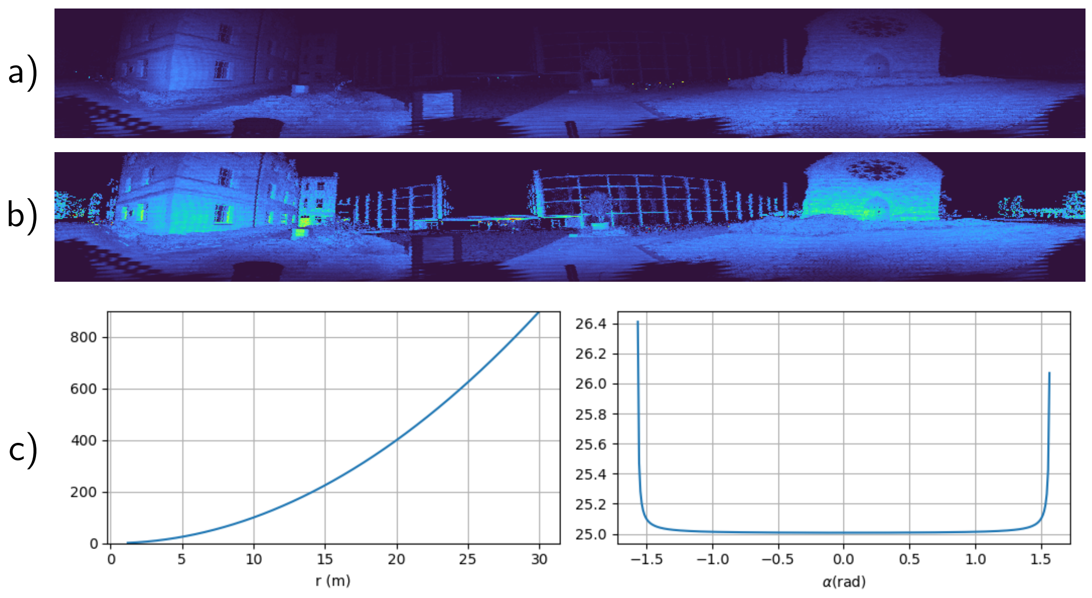

Figure 4.3: Intensity clamped to $[0, 0.3]$ and b) compensation intensity clamped to $[0, 2.5e - 2 \cdot I_{max}]$ using the multiplicative weighted range and angle model with $m = 3546.099$. Noisy angle measurements are better compensated and ground areas are more even. Walls are still uneven. c) shows the model plotted over range (left) and angle (range).

induce further effects inside the sensor.

As proposed for example by Jutzi and Gross [Jutzi and Gross 2009] the relationship might be exponential. This assumption leads to the updated Weighted Range and Angle model. Let w_r and w_α be the model parameters:

$$g(\mathbf{p}; [w_r, w_\alpha]^\intercal) = I \cdot r^{w_r} \cos^{w_\alpha}(\alpha). \tag{4.9}$$

Note´ that negative values are allowed for w_α thereby still following Eq. 2.10. Figure 4.4 shows the resulting compensated intensity. Again some areas like the ground are evenly compensated, while especially walls closer to the sensor are still not evenly compensated. The close range area below $\approx 4m$ is very dark compared to points further away.

4.3.3 Sensor Specific Compensation

The aforementioned compensation approach should work independent of the utilized sensor. As can be seen in Fig. 4.5 the intensity for points close to the sensor increases from about 1 to 4 meters, before decreasing with growing distance. Simply applying model Eq. 4.9 to these points will further reduce this point intensity.

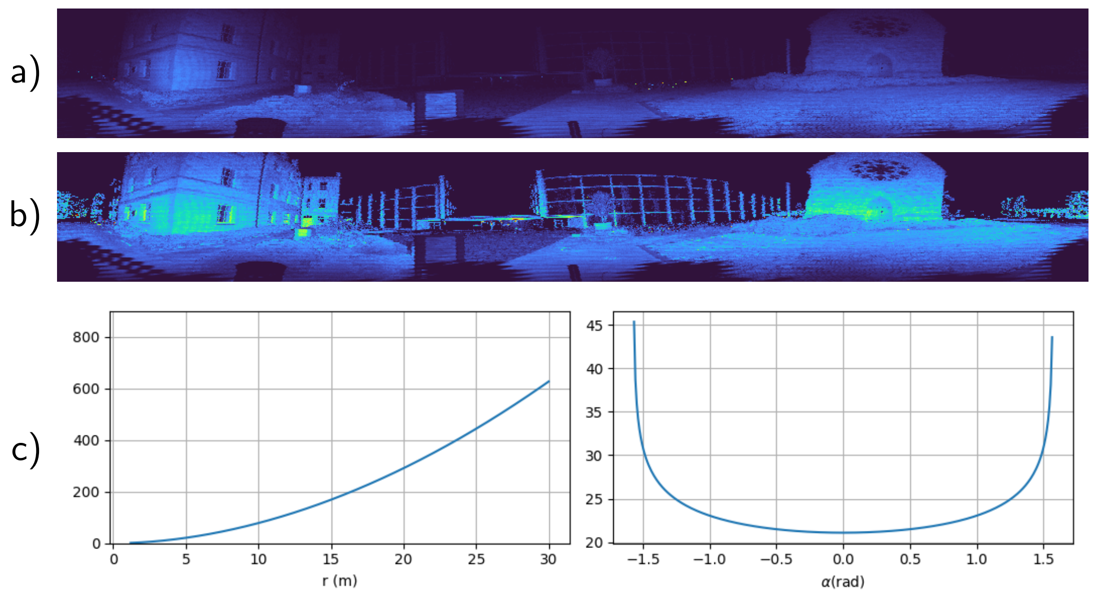

Figure 4.4: a) Intensity clamped to $[0, 0.3]$ and b) compensation intensity clamped to $[0, 0.4]$ using the exponential model with $w_r = 0.947057$ and $w_\alpha = -0.14458$. Areas further from the sensor are show more even intensity. The close range area is still very dark.

This behavior is unexpected with respect to Eq. (2.10). As this effect is consistent between all datasets it is probably sensor related. Sensor specific compensation is necessary. Eq. 2.6 suggests that the relationship between geometric compensation and sensor intrinsic compensation is multiplicative. The sensor model used to record the Newer College Dataset [L. Zhang, Camurri, and Fallon 2021] is the Ouster OS0. Points of distance smaller 1 meter are primarily points generated by the frame holding the sensor and the person moving the frame. As the amount of points at this range is small it is excluded from the model and parameter optimization.

4.3.4 Near Range Cos4

The initial assumption is made, that the reduction may be due to aperture construction or similiar optical effects. Accordingly a $\cos^4$ function is tested. As can be seen in Fig. 4.5 the intensity distribution on different datasets roughly follows the tested function. Accordingly $\cos^4$ is applied to compensate close range points.

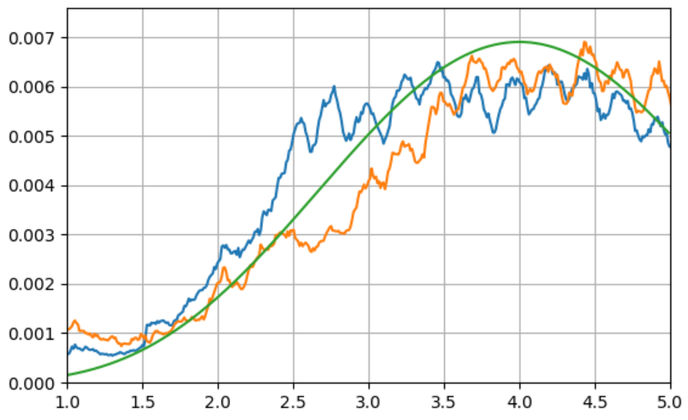

Figure 4.5: Intensity of Newer-College Math-Medium (blue) and Newer-College Quad-Easy (yellow) plotted against r compared to $c_{cos}(\cdot, [0, 4]^\intercal) \cdot 0.0069$ (green) against r.

Given point $\mathbf{p}_i$ and parameters $[r_{\text{min}}, r_{\text{mid}}]^\intercal$:

$$c_{cos}(\mathbf{p}_i; [r_{\text{min}}, r_{\text{mid}}]^\intercal) = \cos^4\left(\frac{r_{\text{mid}} - r_i}{r_{\text{mid}} - r_{\text{min}}} \cdot \frac{\pi}{2}\right), \tag{4.10}$$

$$c_{sys}(\mathbf{p}_i; [r_{\text{min}}, r_{\text{mid}}]^\intercal) = \begin{cases} \frac{1}{c_{cos}} & \text{if } r_i < r_{mid} \\ 1.0 & \text{otherwise} \end{cases}. \tag{4.11}$$

The initial assumption states that this is part of sensor intrinsics η_{sys}. Hence this compensation is combined multiplicatively to the geometric compensation models:

$$g(\mathbf{p}; \phi, [r_{\text{min}}, r_{\text{mid}}]^\intercal) = g(\mathbf{p}; \phi) \cdot c_{sys}(\mathbf{p}; [r_{\text{min}}, r_{\text{mid}}]^\intercal). \tag{4.12}$$

The resulting model compensates the close range area well including the area where the c_{sys} case switches. Fig. 4.6 shows an example of this.

4.3.5 Wave Compensation

Close inspection of the data in Sec. 6.1 shows that a wavelike noise permeates all datasets. This expresses itself in a sinusoidal function dependent on range and intensity magnitude. This is not compensated for in the reflectivity channel either.

As the wave noise seems to scale with intensity, another multiplicative factor is added to the compensation. The wave can be defined by the wave offset ψ, the

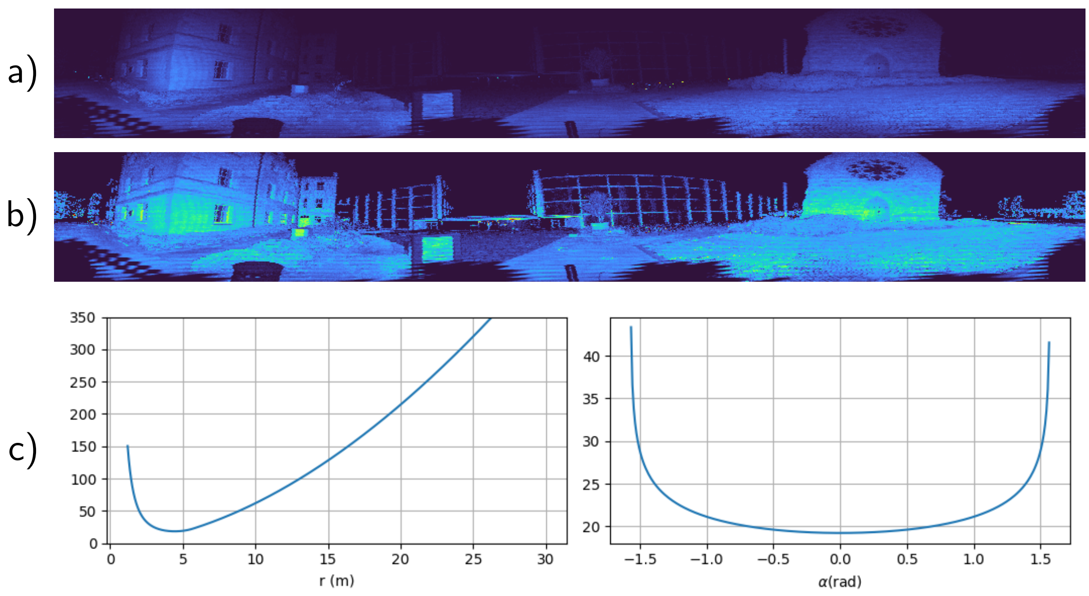

Figure 4.6: a) Intensity clamped to $[0, 0.3]$ and b) compensation intensity clamped to $[0, 0.7]$ using the exponential model with $w_r = 0.895899$ and $w_\alpha = -0.153564$ and c_{sys} with $r_{mid} = 5.67397$ and $r_{min} = 0.0774244$. The close range areas are now compensated for and align evenly with adjacent similar surfaces.

wave length λ and amplitude a:

$$c_{wave}(\mathbf{p}_{\boldsymbol{i}}; [\psi, \lambda, a]^\intercal) = \left(1 + a \cdot \sin\left(\frac{r_i}{\lambda} + \psi\right)\right)^{-1} \tag{4.13}$$

$$g(\mathbf{p}; \phi, [\psi, \lambda, a]^\intercal) = g(\mathbf{p}; \phi) \cdot c_{wave}(\mathbf{p}; [\psi, \lambda, a]^\intercal). \tag{4.14}$$

4.3.6 Vignette Compensation

The last scan system based effect observed seems to be a vignetting effect observed towards the upper and lower scanlines. This may be in part due to geometric effects, as areas close to the lower edge of the scan contain primarily close floor elements. While points in upper rings are often observed from steeper angles and larger range.

As the lower and higher rings of given scan tend to be comparatively dark even after compensation, vignette compensation is explored. Specifically the same 6th-order polynomial proposed by Goldman and Chen [Goldman and Chen n.d.].

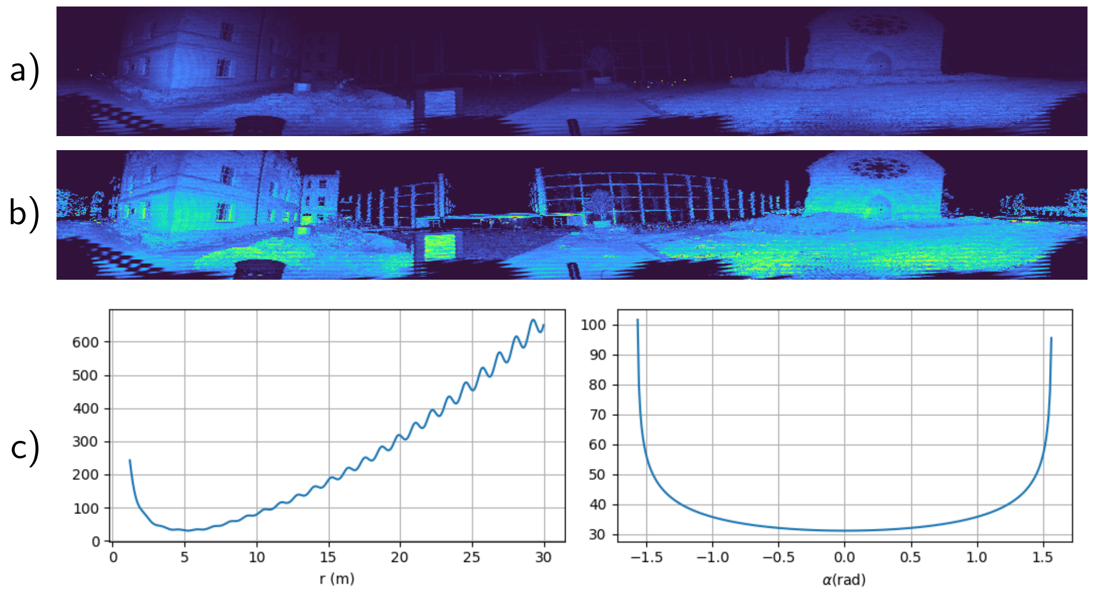

Figure 4.7: a) Intensity clamped to $[0, 0.3]$ and b) compensation intensity clamped to $[0, 1.0]$ using the exponential model and cos4 model with $w_r = 0.955925$ and $w_\alpha = -0.222957$, c_{sys} with $r_{mid} = 6.88915$ and $r_{min} = -0.0525837$ and c_{wave} with $[\psi = 2386.23, \lambda = 0.186251, a = 0.0463039]$.

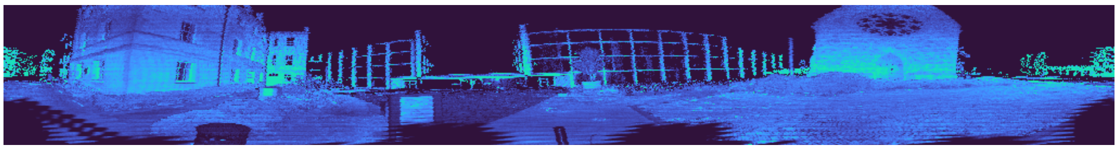

Figure 4.8: Compensated intensity using the exponential model $[w_d = 1.18917, w_\alpha = -0.190482]$, cos4 $[r_{mid} = 4.68113, r_{min} = 0.844142]$ and vignette compensation $[v_1 = 0.163114, v_2 = -2.80794, v_3 = 3.38404]$.

Given point $\mathbf{p}_i$, parameters $[v_1, v_2, v_3]$ and a scan with h rings:

$$\bar{\nu} = 2 \cdot \frac{\nu_i}{h} - 1, \tag{4.15}$$

$$c_{\text{vign}}(\mathbf{p}_i; [v_1, v_2, v_3]) = I_i \cdot (1 + v_1\bar{\nu}^2 + v_2\bar{\nu}^4 + v_3\bar{\nu}^6). \tag{4.16}$$

Compensation using such a model is shown in Fig. 4.8. The polynom parameters vary more widely during optimization.

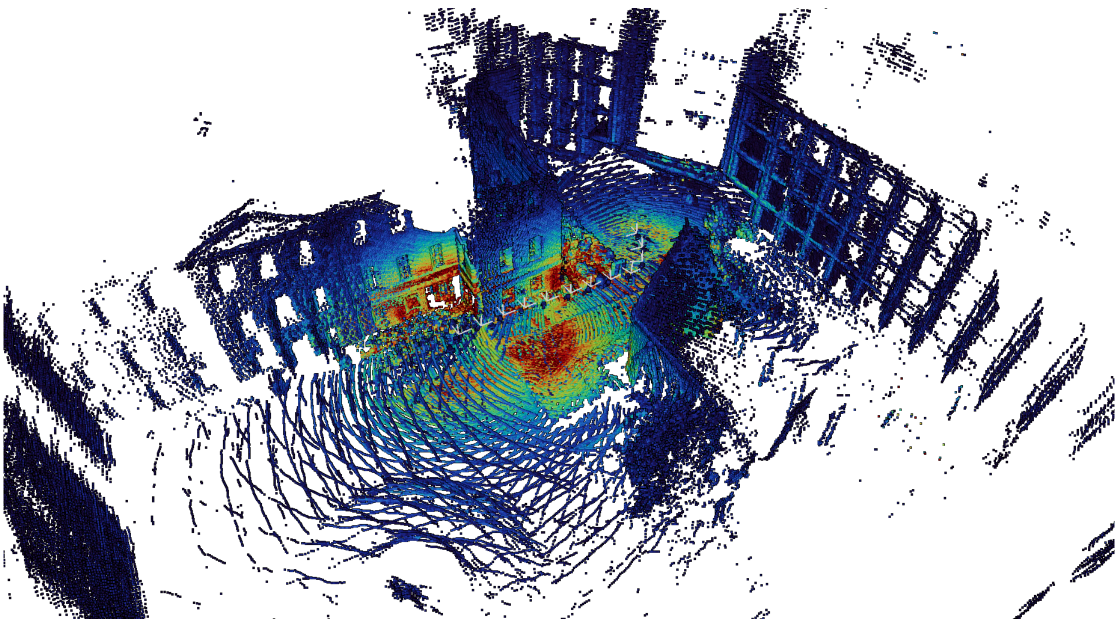

Figure 4.9: Fused point cloud consisting of 12 single point clouds. Also shown are the sampled poses. The minimal distance between poses is $2m$

4.4 Compensation Parameter Optimization

To train the models, points are taken from preregistered and fused point clouds. FastLIO2 [Xu, Cai, et al. 2022] was used to estimate an initial trajectory. One point cloud is taken every roughly $2m$ along the trajectory. The resulting point clouds are fused using the estimated poses, an example of such a point cloud can be seen in Fig. 4.9.

The fused pointcloud $\hat{\mathcal{P}}$ should contain points taken of the same scene objects from different angles and distances. This allows to correlate intensity values taken from different angles and distances and subsequently to estimate parameters of given compensation models presented in chapter 4.2.1. For each point the k-nearest-neighbors are found in $\hat{\mathcal{P}}$ using the method described in Section 4.1.1. This is then used to find incidence angle α. In practice $k = 51$ worked well.

4.4.1 Point Filter

Similar to Section 5.2.1 the input point cloud is prefiltered. The filtered points should allow for stable parameter optimization. Accordingly, step one filters points $\mathbf{p}_i$ based on e and α_i. The second filter again filters outliers using the method described in Section 5.2.1.

Points $\mathbf{p}_i$ with $r_i < 1m$ include sensor frame and operator. As such points are not used in localization and are usually observed from the same orientation they are excluded from model optimization. For models without cos4 close range compensation this minimum range is increased to $r_i \geq 4.5$. Points with $r_i > 20$ are also excluded. This is due to sensor noise increasing over longer distance combined with pose estimation inaccuracies affecting longer range points more. $\mathbf{p}_i$ with $\alpha_i > 1.5$ are also filtered. Small changes in large α_i due to noise or inaccuracy in pose estimation may result in large changes in compensation. This also filters points with invalid estimates of α_i.

The optimization is based on the principle in Eq. 4.6. This is extended to point-pairs that belong to the same surface. As no point cloud segmentation is available local point homogeneity is estimated by the following filter. Given point $\mathbf{p}_i$ and its k-nearest neighbors:

$$\tau_i = \frac{1}{k}\sum_{j=1}^{k}\frac{(I_i - I_j)^2}{(r_i - r_j)^2}. \tag{4.17}$$

For points with small differences in I this value is small. Point pairs with large difference in I and r this value is also small, but grows rapidly with decreasing difference in r.

This value allows for points observed from stronger diverging poses to still be included despite large differences in I, while points observed from similar positions, should have similar I. More research into this filter stage might be of interest.

After these filters each valid point is added to the optimization. A point is valid, if the point itself and all its found neighbors pass all filter steps. Figure 4.10 shows how the parameter based filter followed by the similarity filter change the points used for parameter optimization.

For each $\mathbf{p}_i$ and its neighbor $\mathbf{p}_j$ a point pair is added to the optimization, if:

$$|r_i - r_j| > \Delta_r \tag{4.18}$$

or

$$|\alpha_i - \alpha_j| > \Delta_\alpha. \tag{4.19}$$

This removes point pairs with no difference in parameters, as such points either have no difference in intensity or introduce scanning noise which may lead to parameter overestimation. Earlier tests applied both simultaneously.

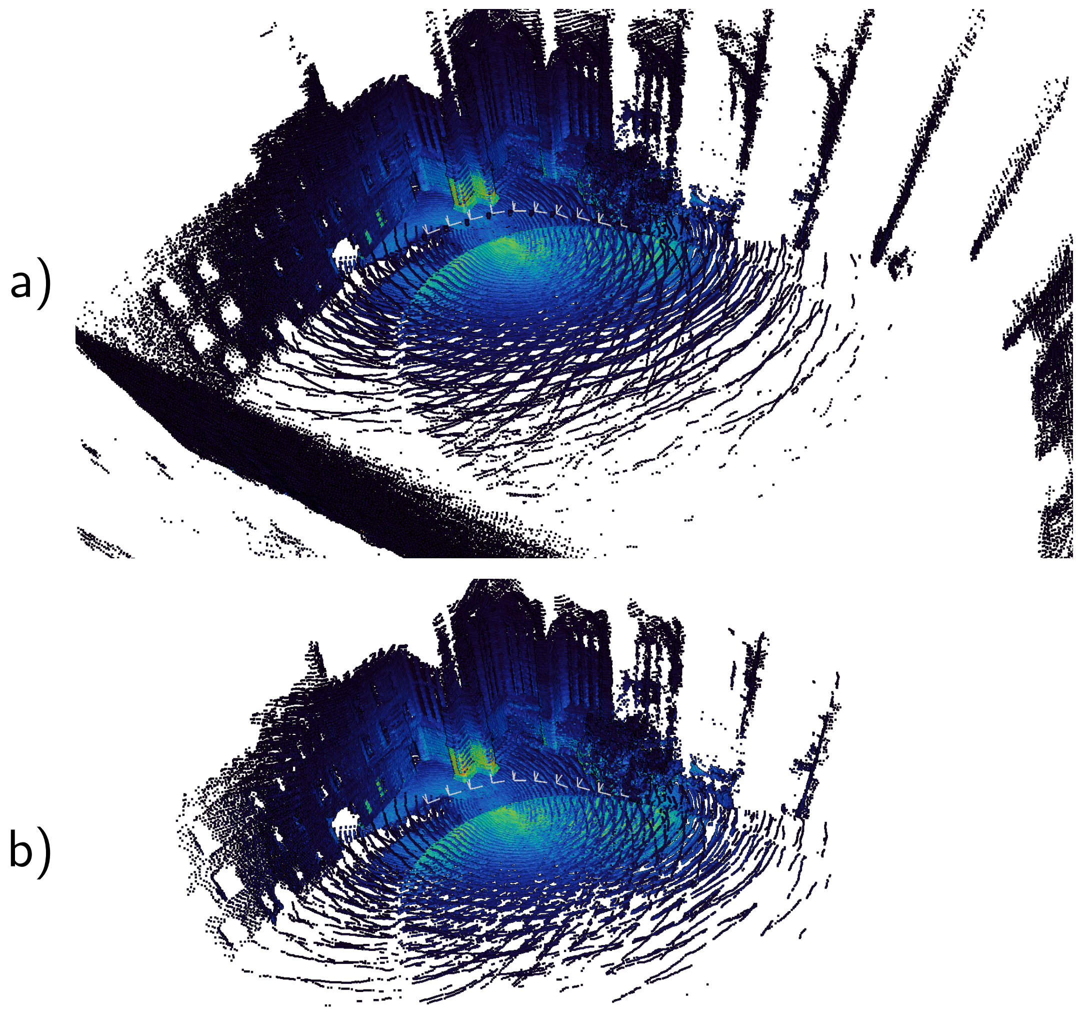

Figure 4.10: a) shows the initial fused point cloud colored by intensity. b) shows the points after the point fiters are applied. Poses are drawn in white. Of the initial 1008092 points 611265 remain. Close range points around the poses are removed as well as points along the outer walls. The point set remaining is the corner inside the trajectory.

4.4.2 Model Parameter Optimization

The criterion in Eq. 4.6 is used to define the loss function $\mathcal{L}$. Two types of loss function are tested. For given point pair $\mathbf{p}_i, \mathbf{p}_j$ and compensation model $g(\cdot; \phi)$, the pairwise loss can be defined as:

$$\mathcal{L}_{ij}(\phi) = \frac{1}{c(\mathbf{p}_i)} \mathcal{L}(\mathbf{p}_i, \mathbf{p}_j; \phi)). \tag{4.20}$$

Two specific loss functions are tested, see Eqs. 4.24 and 4.25.

I magnitude and accordingly $\mathcal{L}$ magnitude is directly proportional to range. Resulting scans have areas of consistently higher I which may skew optimization. The factor $\frac{1}{c(\mathbf{p}_i)}$ is introduced to reweigh loss elements and increase the effect of low I areas and point pairs. Without this factor model parameters are consistently vastly overestimated and results in non-funtional models.

This loss function is minimized if a point and all its neighbors have the same I after compensation. As this chains trough every point and their neighbors, this should lead to non-neighboring points along the same surface to be compensated for the same intensity target.

As this error function depends on pair-wise compensated intensity, a zero function would be an optimal solution. This is addressed by adding a normalization term:

$$\mathcal{L}_n(i; \phi) = \frac{1}{c(\mathbf{p}_i)} (c(\boldsymbol{p}_i) - I_i). \tag{4.21}$$

The used value for $w_n = 0.1$.

The resulting minimization problem is described by:

$$\boldsymbol{e}_{ij} = \begin{pmatrix} \mathcal{L}_{ij}(\phi) \\ w_n \mathcal{L}_n(i; \phi) \end{pmatrix}, \tag{4.22}$$

$$\min_{\phi} = \sum_i \sum_{j=1}^{k} \boldsymbol{e}_{ij}^{\intercal} \boldsymbol{e}_{ij}. \tag{4.23}$$

The resulting problem is solved using the ceres solver [Agarwal, Mierle, and Team 2022]. Specifically the Levenberg-Marquadt [Levenberg 1944 or Marquardt 1963] algorithm for solving non-linear-least-square problems is employed.

Loss Functions

The first loss function used compares the current point $\mathbf{p}_i$ against the precomputed compensation $\hat{\rho}_j$ for the neighboring points $\mathbf{p}_j$:

$$\mathcal{L}_{1,ij}(\phi) = (g(\mathbf{p}_i; \phi) - \hat{\rho}_j). \tag{4.24}$$

This function is optimized by running the optimizer for 4 iterations. Then the compensation $\hat{\rho}$ for all points is computed. This process is redone multiple times. Optimizing using this function proved to be difficult. Convergence happened very slowly.

The second loss function directly compares compensation values:

$$\mathcal{L}_{2,ij}(\phi) = (g(\mathbf{p}_i; \phi) - g(\mathbf{p}_j; \phi)). \tag{4.25}$$

As the compensation results are compared directly, the optimizer is run 30 iterations. 6 restarts are used to reset the optimizer.

5 Localization using FastLIO With Compensated LiDAR Data

5.1 FastLIO2

The localization algorithm used is FastLIO2 [Xu, Cai, et al. 2022] as it is a fast, state of the art localization algorithm that uses only IMU and LiDAR data. It can therefor be adjusted to also take LiDAR intensity or reflectivity data. The following chapter will give an overview over FastLIO2. The formulas in this section are taken from Xu, Cai, et al. 2022, unless otherwise specified. Sec. 5.2 describes the changes added in this work to accommodate further LiDAR data into the FastLIO framework.

FastLIO [Xu and F. Zhang 2021] seeks to create a 3D map of the agents environment while simultaneously moving within this environment and localizing itself. Compared to SLAM algorithms no loop-closure is done. The framework is also capable of estimating the system extrinsic between IMU and LiDAR scanner. This feature is not used in this work as the used dataset provides the extrinsic.

FastLIO2 uses an iterated Kalman-Filter for agent localization. This is done using the positional information given by IMU, combined with correction steps using the LiDAR scans. The IMU generates odometry information at a much faster rate than the LiDAR generates full point clouds. FastLIO accumulates IMU data during the filters forward step. Once a LiDAR scan is accumulated the backward step is initiated. Fig. 5.1 shows the major components of the FastLIO2 localization algorithm.

5.1.1 FastLIO State Estimation

The FastLIO Kalman filter state consists of $\boldsymbol{R}_I$, $\boldsymbol{p}_I$ and v_I the IMU rotation and position compared to the global frame and velocity. b_a and b_ω are the IMU measurement noises. g is the current gravity vector. The last two elements R_L and p_L are extrinsic estimations. The resulting state stems from the manifold $\mathcal{M} = SO(3) \times \mathbb{R}^{15} \times SO(3) \times \mathbb{R}^3$.

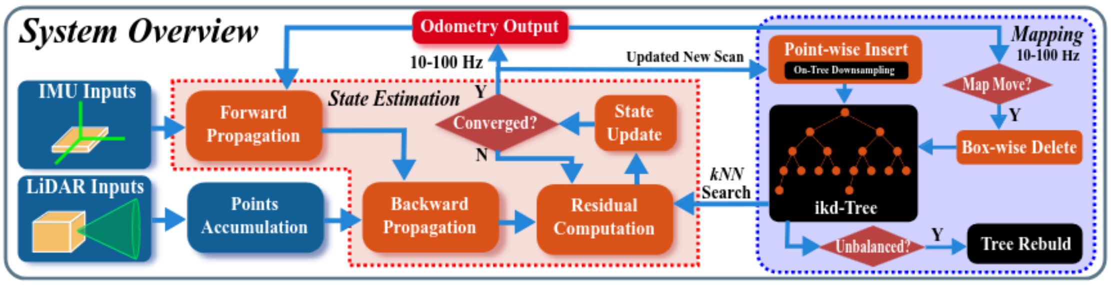

Figure 5.1: Figure taken from Xu, Cai, et al. 2022 Fig. 1. Overview over the FastLIO2 algorithm. The algorithm consists of two major elements. First the iterated Kalman Filter used for state estimation and second the ikd-Tree used for quickly adding and looking up map points.

The forward input u consists of the IMU rotation measurement ω and position a with noise $\boldsymbol{b}_\omega$ and $\boldsymbol{b}_a$. The noise is modeled as a random walk governed by variables n_ω and n_a. The noise of the extrinsic parameters is controlled by parameters $\boldsymbol{n}_{b\omega}$ and $\boldsymbol{n}_{ba}$. This creates the noise vector w

The resulting state transition function f is defined as:

$$x = [\boldsymbol{R}_I^\intercal, \boldsymbol{p}_I^\intercal, \boldsymbol{v}_I^\intercal, \boldsymbol{b}_\omega^\intercal, \boldsymbol{b}_a^\intercal, \boldsymbol{g}^\intercal, \boldsymbol{R}_L^\intercal, \boldsymbol{p}_L^\intercal] \in \mathcal{M}, \tag{5.1}$$

$$u = [\omega^\intercal, \boldsymbol{a}^\intercal], \tag{5.2}$$

$$w = [\boldsymbol{b}_\omega^\intercal, \boldsymbol{b}_a^\intercal, \boldsymbol{n}_{b\omega}^\intercal, \boldsymbol{n}_{ba}^\intercal], \tag{5.3}$$

$$f(x, u, w) = \begin{pmatrix} \omega - b_\omega - n_\omega \\ \boldsymbol{v}_I + \frac{1}{2}(R_I(\boldsymbol{a} - \boldsymbol{b}_a - \boldsymbol{n}_a) + \boldsymbol{g})\Delta t \\ R_I(\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) + g \\ \boldsymbol{n}_{b\omega} \\ \boldsymbol{n}_{ba} \\ \boldsymbol{0} \in \mathbb{R}^9 \end{pmatrix} \in \mathbb{R}^{24}.$$

The state propagation can now be defined using the $\boxplus$ and $\boxminus$ operators introduced by Hertzberg et al. [Hertzberg et al. 2013]. These operators ensure that the results of the operations stay on the manifold, thereby eliminating the need for renormalizing the results. Specifically let $\mathcal{M}$ be a manifold on $\mathbb{R}^n$:

$$\boxplus : \mathcal{M} \times \mathbb{R}^n \to \mathcal{M} \tag{5.4}$$
$$\boldsymbol{X} \boxplus \boldsymbol{a} = \boldsymbol{X} \exp(\boldsymbol{a}) \qquad \text{if } \boldsymbol{X} \in SO(3), \boldsymbol{a} \in \mathbb{R}^3, \tag{5.5}$$
$$\boldsymbol{a} \boxplus \boldsymbol{b} = \boldsymbol{a} + \boldsymbol{b} \qquad \text{if } \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3, \tag{5.6}$$
$$\boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n \qquad , \tag{5.7}$$
$$\boldsymbol{X}_1 \boxminus \boldsymbol{X}_2 = \boldsymbol{X}_1 \log(\boldsymbol{X}_2) \qquad \text{if } \boldsymbol{X}_1, \boldsymbol{X}_2 \in SO(3), \tag{5.8}$$
$$\boldsymbol{a} \boxminus \boldsymbol{b} = \boldsymbol{a} - \boldsymbol{b} \qquad \text{if } \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3. \tag{5.9}$$

These operators use the specialized exp and log functions described in by Hertzberg et al. [Hertzberg et al. 2013]. Given input vector $\boldsymbol{a}$ and $\boldsymbol{X} \in SO^3$:

$$h = \frac{1 - \cos|\boldsymbol{a}|}{|\boldsymbol{a}|^2}, \tag{5.10}$$
$$s = \operatorname{sinc}|\boldsymbol{a}|, \tag{5.11}$$
$$\exp(\boldsymbol{a}) = \exp\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos|\boldsymbol{a}| + hx^2 & -sz + hxy & sy + hxz \\ sz + hxy & \cos|\boldsymbol{a}| + hy^2 & -sx + hyz \\ -sy + hxz & sx + hyz & \cos|\boldsymbol{a}| + hz^2 \end{pmatrix}, \tag{5.12}$$
$$ac(\boldsymbol{X}) = \operatorname{acos}\frac{tr(\boldsymbol{X}) - 1}{2}, \tag{5.13}$$
$$\log \boldsymbol{X} = \frac{ac(\boldsymbol{X})}{2 \sin ac(\boldsymbol{X})}\begin{pmatrix} \boldsymbol{X}_{32} - \boldsymbol{X}_{23} \\ \boldsymbol{X}_{13} - \boldsymbol{X}_{31} \\ \boldsymbol{X}_{21} - \boldsymbol{X}_{12} \end{pmatrix}. \tag{5.14}$$

The uncertainty at timestep i is comuted using the covariance $\boldsymbol{Q}_i$ of the noise vector $\boldsymbol{w}$ combined with the matrices $\boldsymbol{F}_{\boldsymbol{x}_i}$ and $\boldsymbol{F}_{\boldsymbol{w}_i}$ defined in FastLIO [Xu and F. Zhang 2021] and derived by Hertzenberg et al. [Hertzberg et al. 2013]. Putting these operations together with the current state and update functions results in the state and covariance update function at timestep i:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i \boxplus \Delta t \cdot f(\boldsymbol{x}_i, \boldsymbol{w}_i, \boldsymbol{u}_i), \tag{5.15}$$
$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i \boxminus \boldsymbol{x}_k \tag{5.16}$$
$$\boldsymbol{F}_{\tilde{\boldsymbol{x}}_i} = \left.\frac{\delta(\boldsymbol{x}_{i+1} \boxminus \hat{\boldsymbol{x}}_{i+1})}{\delta\tilde{\boldsymbol{x}}}\right|_{\tilde{\boldsymbol{x}}_i=0, \boldsymbol{w}_i=0}, \tag{5.17}$$
$$\boldsymbol{F}_{\boldsymbol{w}_i} = \left.\frac{\delta(\boldsymbol{x}_{i+1} \boxminus \hat{\boldsymbol{x}}_{i+1})}{\delta\boldsymbol{w}}\right|_{\tilde{\boldsymbol{x}}_i=0, \boldsymbol{w}_i=0}. \tag{5.18}$$

Assuming that $\mathcal{P}_k$ is the last point cloud optimized for. Let $\hat{\boldsymbol{x}}_k$ be the state and $\hat{\boldsymbol{P}}_k$ the covariance matrix directly after this point cloud was incorporated into the state. Forward propagation is done whenever new IMU measurements are received:

$$\begin{align}
\boldsymbol{x}_0 &= \hat{\boldsymbol{x}}_k, \tag{5.19}\\
\boldsymbol{P}_0 &= \hat{\boldsymbol{P}}_k, \tag{5.20}\\
\boldsymbol{x}_{i+1} &= \boldsymbol{x}_i \boxplus \Delta t \cdot f(\boldsymbol{x}_i, \boldsymbol{w}_i, \boldsymbol{u}_i), \tag{5.21}\\
\boldsymbol{P}_{i+1} &= \boldsymbol{F}_{\tilde{\boldsymbol{x}}_i} \boldsymbol{P}_i \boldsymbol{F}_{\tilde{\boldsymbol{x}}_i}^{\intercal} + \boldsymbol{F}_{\tilde{\boldsymbol{w}}_i} \boldsymbol{Q}_i \boldsymbol{F}_{\tilde{\boldsymbol{w}}_i}^{\intercal}. \tag{5.22}
\end{align}$$

These operations are repeated until a new LiDAR scan is received or accumulated.

Residual Computation

To integrate the LiDAR information into the state estimate, a subset of the received point cloud is used. Sec. 5.2 describes the selection criteria for both the original algorithm and the updated version used to include intensity data into FastLIO. Let $\mathcal{P} = \{\mathbf{p}_0, \ldots, \mathbf{p}_m\}$ be the set of selected points translated to the global frame using the current position estimate $\boldsymbol{p_I}$ and rotation estimate $\boldsymbol{R_I}$.

For each of point $\boldsymbol{p_j}$ the k nearest points in the current map are found. The assumption is made that each input point should lie on a plane generated by their nearest neighbors. Specifically let $\boldsymbol{n}_o$ be the normal and $\boldsymbol{p}_o$ be the origin of the plane generated, $\boldsymbol{T_{I_k}}$ the current pose estimate, $\boldsymbol{T_{L_k}}$ the current extrinsic estimate and $\boldsymbol{n}_j$ the estimated measurement noise:

$$\begin{align}
d_o(\boldsymbol{p}) &= \boldsymbol{n}_o^{\intercal}(p - p_o), \tag{5.23}\\
\hat{\boldsymbol{p}}_{i,j} &= \boldsymbol{T_{I_k}} \boldsymbol{T_{L_k}} \boldsymbol{p}_{i,j}, \tag{5.24}\\
z_i &= d_o(\hat{\boldsymbol{p}}_{i,j}). \tag{5.25}
\end{align}$$

For each point for which sufficient neighbors can be found this residual can be computed. This error is than approximated by its first order approximation. Let $\boldsymbol{H}_j$ be the Jacobian matrix:

$$\begin{align}
h_j(\boldsymbol{x}) &\approx h_j(\hat{\boldsymbol{x}}) + H_j \widetilde{\boldsymbol{x}} + \delta_j, \tag{5.26}\\
H_j &= \frac{\partial h_j\left(\hat{\boldsymbol{x}} \boxplus \widetilde{\boldsymbol{x}}\right)}{\partial \widetilde{\boldsymbol{x}}}\bigg|_{\widetilde{\boldsymbol{x}}=\mathbf{0}}, \tag{5.27}\\
\boldsymbol{z}_j &= h_j(\hat{\boldsymbol{x}}). \tag{5.28}
\end{align}$$

As the residual $\boldsymbol{z}_j$ is now independent of measurement noise, it can be computed directly.

Given the state estimate $\hat{\boldsymbol{x}}_\kappa$ after the κ-th iteration of the iterated Kalman filter, the error between the current propagated state $\hat{\boldsymbol{x}}$ and the unknown optimal state $\boldsymbol{x}$ is defined as:

$$\begin{aligned} \tilde{\boldsymbol{x}}_\kappa &= \boldsymbol{x} \boxminus \hat{\boldsymbol{x}}_\kappa, && (5.29)\\ \boldsymbol{x} \boxminus \hat{\boldsymbol{x}} &= (\hat{\boldsymbol{x}}_\kappa \boxplus \tilde{\boldsymbol{x}}_\kappa) && (5.30)\\ &= \hat{\boldsymbol{x}}_\kappa \boxminus \hat{\boldsymbol{x}} + \boldsymbol{J}_\kappa \tilde{\boldsymbol{x}}_\kappa \sim \mathcal{N}(0, \hat{\boldsymbol{P}}), && (5.31)\\ \boldsymbol{J}_\kappa &= \begin{pmatrix} \boldsymbol{A}(\hat{\boldsymbol{R}}_{I,\kappa} \boxminus \hat{\boldsymbol{R}}_{I_k})^{-\intercal} & \boldsymbol{0}_{3\times15} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{15\times3} & \boldsymbol{I}_{15\times15} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times15} & \boldsymbol{A}(\hat{\boldsymbol{R}}_{L_k,\kappa} \boxminus \hat{\boldsymbol{R}}_L)^{-\intercal} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times15} & \boldsymbol{0}_{3\times3}\boldsymbol{I}_{3\times3} & \end{pmatrix}. && (5.32) \end{aligned}$$

The $A()^{-1}$ is again introduced by Hertzberg et al. [Hertzberg et al. 2013].

Combining this state estimation problem with the residuals estimated in (5.26), constitutes the following maximum a-posteriori estimation problem:

$$\min_{\tilde{x}_\kappa} \left(||\boldsymbol{x} \boxminus \hat{\boldsymbol{x}}||^2_{\hat{\boldsymbol{P}}} + \sum_{j=1}^{m} ||z_{j,\kappa} + H_j \tilde{\boldsymbol{x}}||^2_{\boldsymbol{R}_j} \right). \tag{5.33}$$

The problem (5.33) can be solved by the following iterated Kalman filter introduced in FastLIO [Xu and F. Zhang 2021]:

$$\begin{aligned} \boldsymbol{H} &= (\boldsymbol{H}_{1,\kappa}, \dots, \boldsymbol{H}_{m,\kappa})^\intercal, && (5.34)\\ \boldsymbol{R} &= diag(\boldsymbol{R}_1, \dots, \boldsymbol{R}_m), && (5.35)\\ \boldsymbol{P} &= (\boldsymbol{J}_\kappa)^{-1} \hat{\boldsymbol{P}}_k (\boldsymbol{J}_\kappa)^{-\intercal}, && (5.36)\\ \boldsymbol{z}_{k,\kappa} &= (\boldsymbol{z}^\intercal_{1,\kappa}, \dots, \boldsymbol{z}^\intercal_{m,\kappa})^\intercal, && (5.37)\\ \boldsymbol{K} &= (\boldsymbol{H}^\intercal \boldsymbol{R}^{-1} \boldsymbol{H} + \boldsymbol{P}^{-1})^{-1} \boldsymbol{H}^\intercal \boldsymbol{R}^{-1}, && (5.38)\\ \hat{\boldsymbol{x}}_{\kappa+1} &= \hat{\boldsymbol{x}}_\kappa \boxplus (-\boldsymbol{K}\boldsymbol{z}_\kappa - (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})(\boldsymbol{J}_\kappa)^{-1}(\hat{\boldsymbol{x}}_\kappa \boxminus \hat{\boldsymbol{x}}). && (5.39) \end{aligned}$$

This method only needs to invert the state matrix instead of the measurement matrix. The equations (5.38) and (5.39) are repeated until convergence: $||\hat{\boldsymbol{x}}_{\kappa+1} \boxminus \hat{\boldsymbol{x}}_\kappa|| < \epsilon$.

The resulting state and covariance estimates are:

$$\begin{aligned} \boldsymbol{x}_k &= \hat{\boldsymbol{x}}_{k,\kappa+1}, && (5.40)\\ \boldsymbol{P}_k &= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}. && (5.41) \end{aligned}$$

5.1.2 Mapping using incremental kd-Tree

The main improvement of FastLIO2 over FastLIO is the use of a map based of an incremental kd-Tree. The underlying structure is a binary search tree. Point data is located at both leaf nodes and internal nodes.

Point Insertion

The input point filter partitions the input field into evenly sized cubes. For given input point $\boldsymbol{p}$ the containing cube C is found. Now all points $\boldsymbol{p}_i \in C$ in the ikd-Tree are found. Let $\boldsymbol{p}_C$ be the center of C. Given the set $V = \{\boldsymbol{p}_i, \dots\} \cup \boldsymbol{p}$ of all points contained within C:

$$\boldsymbol{p}_c = \{\boldsymbol{p}_j \in V |\ \nexists(\boldsymbol{p}_i \neq \boldsymbol{p}_j : ||\boldsymbol{p}_C - \boldsymbol{p}_j|| < ||\boldsymbol{p}_C - \boldsymbol{p}_i||)\}. \tag{5.42}$$

This nearest point is the only one kept. If it is not part of Map it is added. All other points are deleted using the BoxwiseDelete operations.

Boxwise Deletion

The ikd-Tree utilizes lazy-deletion. If a point is set to be deleted, only an internal flag is set. Deletion of nodes and branches is only done during tree rebalacing.

To delete elements in given cuboid area, the cube is compared to the border information in each passing node. If the cube is partly within the borders of given node, the search is continued with the child nodes. If the node borders are fully within the search cube, the node is set to be deleted.

Tree Balancing

If after an incremental operation, a subtree is not α-balanced [Truemper 1982] this subtree is rebuild. As the tree may contain nodes set to be deleted, a second criterion is added. A node conformes to the α-deletion criterion, if sufficient child nodes are set to be deleted. If any of these criteria is violated the subtree rooted at the node is rebuild.

First the tree structure is flattened and all points not set for deletion are added to a flat storage array. This array is then used to rebuild a balanced kd-tree. The new root node is then added to the original parent.

As this operation can be time intensive, FastLIO2 uses a parrallelized framework that allows rebuilding during runtime as well as lookup during the rebuilding process. This is achieved by building the rebalanced tree, while keeping the original input tree operational. All changes added to the original tree during rebuilding are

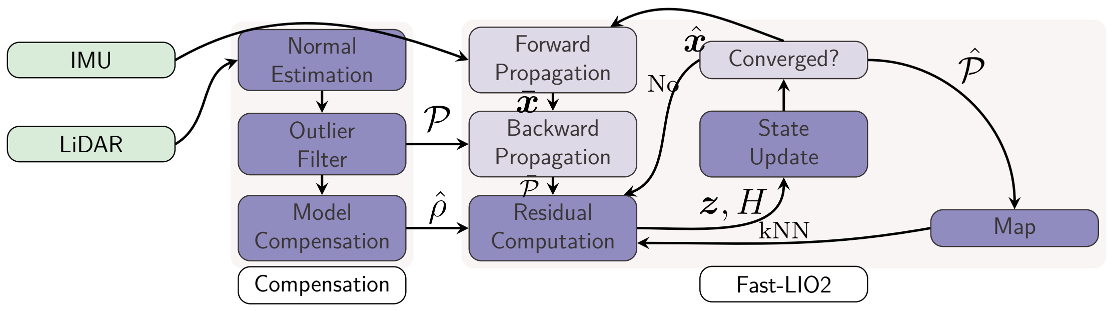

Figure 5.2: The updated FastLIO pipeline preprocesses the input LiDAR point cloud by first estimating point normals and filtering outlier points. All remaining points are intensity compensated. Geometric information of all remaining points in $\mathcal{P}$ are processed like before. The intensity information is used to compute an extra residual term which are then fed to state estimation.

also added to the new tree. Once the new tree is completely build and updated, it is added in place of the original tree.

5.2 Used Additional LiDAR Data

The original FastLIO2 framework only uses geometric information presented by the LiDAR scans. The assumption is made that intensity or further information channels can improve the localization performance of the framework. The proposed addition consists of two parts: First providing FastLIO2 with valid, compensated intensity information and second adding intensity information to the residual computation. In the following section intensity will be used as shorthand for intensity or other LiDAR provided channels.

5.2.1 Point Cloud Preprocessing

To use intensity for localization, intensity should allow to correlate points. Accordingly intensity of a given point should be stable, when observed from a multitude of poses. Neither intensity nor reflectivity of the Newer College datasets sufficiently satisfy this criterion. To further sanitize the input a set of filters are applied.

Given an input point cloud the raw intensity is usually provided as integers. The first stage is to scale the input intensity to unit range floating point.

Point Filter

High gloss surfaces in the scene may create disproportionately large intensity values if hit under small incidence angles. As high gloss areas violate the initial assumption, regarding point intensity, these points are filtered.

To address this issue the assumption is made that intensity values are sampled from a limited distribution X, that can be approximated by the current in scene intensity values. Given intermediate point cloud $\mathcal{P}_n^*$ intensity mean and variance are computed

Given the filter parameter m_{out} the following filter is applied to all $\boldsymbol{p}^* \in \mathcal{P}_n^*$:

$$I_i = \begin{cases} I_i & \text{if } |I_i - \mathbb{E}[X]| < m_{out} \cdot \text{var}[X] \\ 0 & \text{otherwise} \end{cases} . \tag{5.43}$$

The parameter m_{out} has to be chosen, such that as many points as possible are kept throughout the whole scene observed by the agent, while robustly filtering all outlier points. In practice this cutoff is relatively generous as the intensity difference between outliers and normal points tend to be large. Points without valid r or large α are also filtered. Usually $1 < r_i < 20$ and $|\alpha_i| < 1.5$ is chosen.

Incidence Angle Computation

The current point cloud only contains positional, range and intensity information. The first step towards intensity compensation is the incidence angle computation. Hence the per-point, local plane normal has to be estimated.

The point cloud output generated by the Ouster OS0 sensor is ordered in row-major order. Due to sensor construction each row is slightly offset towards the next. Approximations of these offsets are known, although these might change slightly depending on range. To utilize the ordered list for knn-search, first all rows are shifted by their known offset.

Now the order property is used to run the knn-approximation introduced in Sec. 4.1.1, followed by the normal and incidence angle computation. To guarantee approximation accuracy for the normal computation, a minimum of five close neighbors must be found. Otherwise no normal is computed and the corresponding point intensity is set to zero. The resulting output point clouds loose between 40% to 70% of the initial points.

Intensity Compensation

Intensity for any point still valid is compensated using one of the schemes proposed in Chapter 4.3. The resulting point cloud only consists of points with valid compensated intensity.

Implementation

FastLIO2 is implemented to utilize the ROS framework [Quigley et al. 2009]. Accordingly the point cloud preprocessing necessary is implemented in a separate ROS-Node. Given a raw input cloud, first input scaling and outlier filters are applied. Then knn-search is done for all valid points in a multi-thread manner. Followed by the normal and incidence angle computation again using multiple threads. The remaining valid points are then assembled into a new PointCloud message and send to the FastLIO2 node.

5.2.2 Intensity Residuals

To utilize LiDAR intensity data in FastLIO2 the update step has to be augmented by adding intensity residuals for all points.

Given a point $\boldsymbol{p}$, with intensity I, and its k nearest neighbors $\boldsymbol{p}_1, \dots, \boldsymbol{p}_k$ in the current map. The plane $(\boldsymbol{n}_o, \boldsymbol{p}_o)$ is already estimated in the original algorithm. Given rotation $\boldsymbol{R}_{\boldsymbol{n}}$ such that $\boldsymbol{R}_{\boldsymbol{n}}\boldsymbol{n}_o = (0, 0, 1)^\intercal$:

$$\pi(\boldsymbol{p}) = [\boldsymbol{R}_{\boldsymbol{n}}]_{2\times 3}(\boldsymbol{p} - d_o(\boldsymbol{p})\boldsymbol{n}_o). \tag{5.44}$$

This operation projects the point and its neighbors onto the plane and then rotates them such that the plane spans fully along the x- and y-axis. The last row of $\pi(\boldsymbol{p})$ can be dropped as $p_z = -\boldsymbol{n}_o^\intercal \boldsymbol{p}_o$.

Using these points a smooth local intensity function $f(\bar{\boldsymbol{p}})$ parameterized by $\boldsymbol{\lambda}$ is approximated using:

$$\boldsymbol{c}(x, y) = (x^2, y^2, xy, x, y, 1)^\intercal \in \mathbb{R}^6, \tag{5.45}$$

$$f(\bar{\boldsymbol{p}}) = \boldsymbol{\lambda} \cdot c(\bar{\boldsymbol{p}}). \tag{5.46}$$

where x and y are the query point coordinates. The parameter vector $\boldsymbol{\lambda}$ is estimated by solving the following closed form problem (5.50):

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{c}(\bar{\boldsymbol{p}}_1) \\ \dots \\ \boldsymbol{c}(\bar{\boldsymbol{p}}_k) \end{pmatrix} \in \mathbb{R}^{k\times 6}, \tag{5.47}$$

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 3} & \boldsymbol{A}^{\intercal} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{0}_{3\times 3} & \\ \multicolumn{2}{c}{\boldsymbol{A}} & \boldsymbol{0}_{6\times 6} \end{pmatrix} \in \mathbb{R}^{(6+k)\times(6+k)}, \tag{5.48}$$

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{0}_{6\times 1} \\ I_1 \\ \vdots \\ I_k \end{pmatrix} \in \mathbb{R}^{6+k} \tag{5.49}$$

$$\boldsymbol{B\lambda} = \boldsymbol{z} \tag{5.50}$$

This problem is solved using a rank-revealing QR decomposition. The QR decomposition is sufficiently fast and numerically stable to be used in real-time applications.

Using f the local error component is defined as:

$$z_I(\boldsymbol{p}) = f(\pi(\boldsymbol{p})) - I. \tag{5.51}$$

The necessary Jacobian $\nabla z_I(\boldsymbol{p})$ can be found using the chain rule and the jacobians of Eq. 5.44 and Eq. 5.46:

$$\nabla\pi = [\boldsymbol{R_n}]_{2\times 3}(\boldsymbol{I}_{3\times 3} - \boldsymbol{n}\boldsymbol{n}^{\intercal}) \in \mathbb{R}^{2\times 3}, \tag{5.52}$$

$$\nabla f(\bar{\boldsymbol{p}}) = \left[\begin{pmatrix} 2\lambda_1 & \lambda 3 & \lambda_4 \\ \lambda_3 & \lambda_2 & \lambda_5 \end{pmatrix} \begin{pmatrix} \bar{\boldsymbol{p}} \\ 1 \end{pmatrix}\right]^{\intercal} \in \mathbb{R}^{1\times 2} \tag{5.53}$$

$$\nabla z_I = \nabla f(\pi(\boldsymbol{p})) \cdot \nabla\pi(\boldsymbol{p}) \in \mathbb{R}^{1\times 3} \tag{5.54}$$

The gradient scaling factor $||\nabla f||^{-1}$ was found to increase stability independent of input magnitude. This specifically allows easier parameterization independent of input channel. It also has a stabilizing effect on the FastLIO update. This factor might not be ideal and further research to determine better solution is warranted.

As the intensity residuals tend to be large compared to the geometric residuals a weight parameter w_I is introduced. The resulting residuals are then added to the residual set in Eqn. (5.37). Eqn. (5.39) also has to be adjusted.

$$\boldsymbol{z}_i = \begin{pmatrix} \boldsymbol{n}^\mathsf{T}(\boldsymbol{p}_i - \boldsymbol{p}_o) \\ w_I f(\pi(\boldsymbol{p}_i)) - I_i \end{pmatrix}. \tag{5.55}$$

6 Evaluation

6.1 Newer College Dataset

The extension to the Newer College dataset [L. Zhang, Camurri, and Fallon 2021] is a set of combined LiDAR scans, IMU measurement and camera images. The device used to record the data is a handheld device containing an Ouster OS0-128 LiDAR scanner with an ICM-20948 IMU system and 4 Alphasense cameras with a Bosch BM1085 IMU. The extrinsic transformations between the sensors is known and part of the dataset. Of primary interest for this work is the LiDAR scanner and the attached IMU.

Quad

The quad area consists of three sequences recorded in a courtyard consisting of two smaller walls and two longer walls. The yard is surrounded on all sides by 3-story or comparable height brick walls. The ground is paved with an oval spot of grass in the center. Fig. 6.1 shows the intensity and reflectivity values over the quad-easy recording plotted against different measurements.

There are a total of three dataset recorded inside this environment: $Quad-Easy$ consists of two loops around the grass oval at walking pace, $Quad - Medium$ consists again of two loops around the inner court with rotating and swinging motions of the sensor and $Quad - Hard$ consists of one loop of fast walking with fast changes in distance to the walls. This is combined with rapid sensor motion independent of agent movement. The agent enters one of the side buildings.

Overall the localization scenario is constraint primarily by the surrounding walls, the single tree and the central grass oval.

Park

The $Park$ dataset is a long recording of 1567 seconds. The recording starts in the same courtyard as the quad datasets. From there the agent moves into another yard surrounded on three sides by walls. The last area is the actual park. This is a large open areas constrained by small brick walls. The area also contains a lot of trees and smaller vegetation.

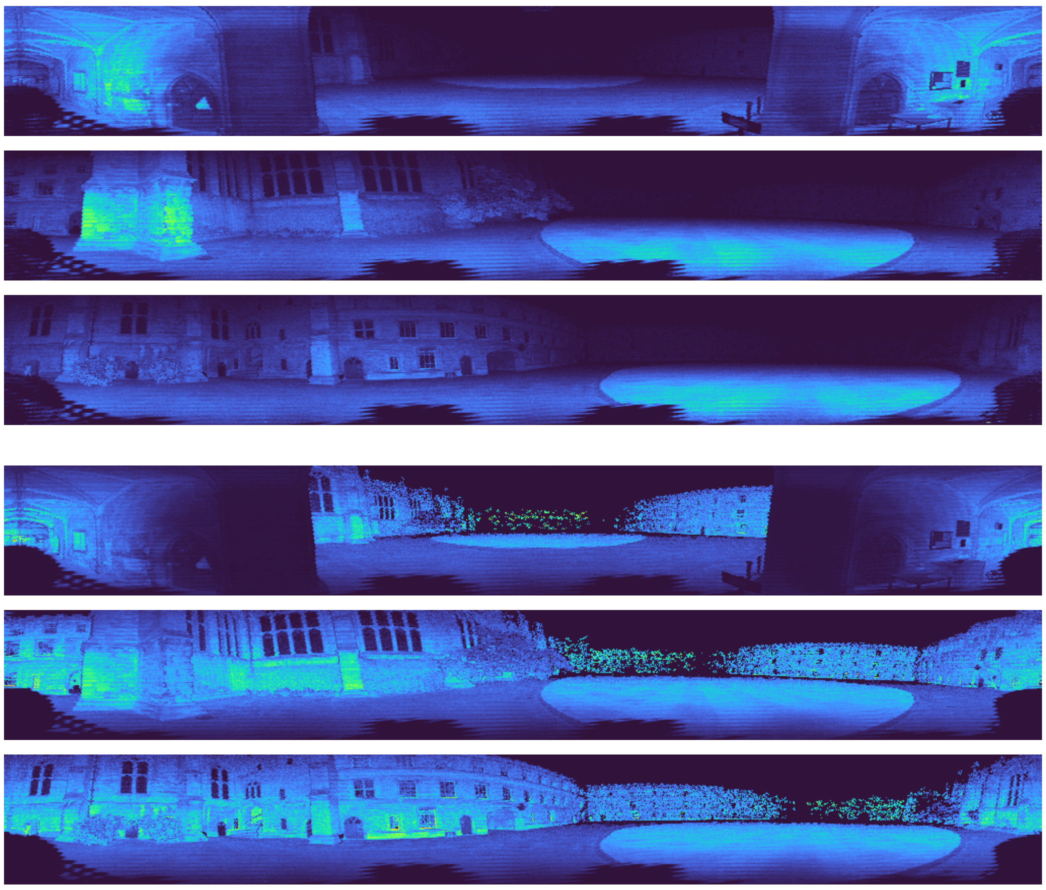

Figure 6.1: Intensity values scaled to [0,0.35] (top) and reflectivity values (bottom) of the quad-easy dataset

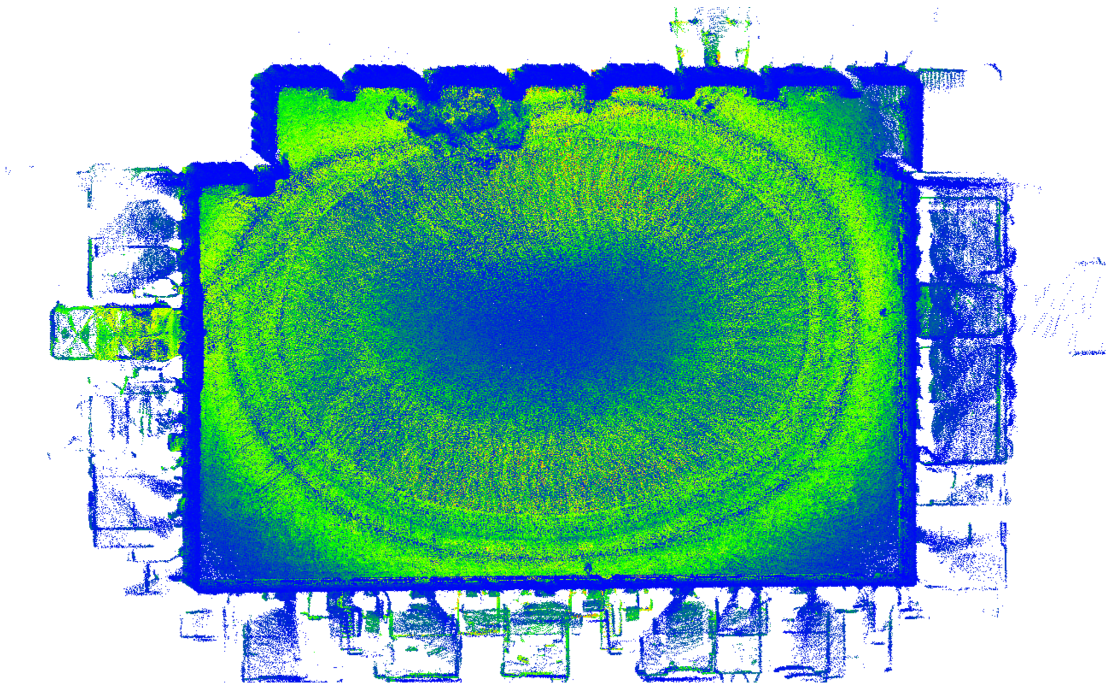

Figure 6.2: Map of the quad-easy dataset colored by intensity

The resulting illumination negatively affects LiDAR intensity measurements. Branches tend to be thin compared to the actual area illuminated by the laser flash. The resulting range measurement tends to be accurate, but the measured intensity consists also of background illumination filling the reflected area not filled by the branch.

Cloister

The *cloister* dataset contains two rounds around the cloister corridor and the inner cloister yard. The corridor consists of the brick wall to one side, a covered paved corridor, that opens to one side into the central yard. Most points recorded are at short range. Later the operator moves into the yard.

Math Institute

This dataset is recorded outside the mathematics institute. It consists of narrow alleys between building combined with a lot reflecting surfaces. It starts and ends on an open square including longer ranges. Buildings in scenes consists of old stone building like a chapel and modern glass-clad buildings. Accordingly this dataset contains a significant amount of non-lambertian surfaces.

Again three sequences are provided. Math-Easy consists of walking pace explo-

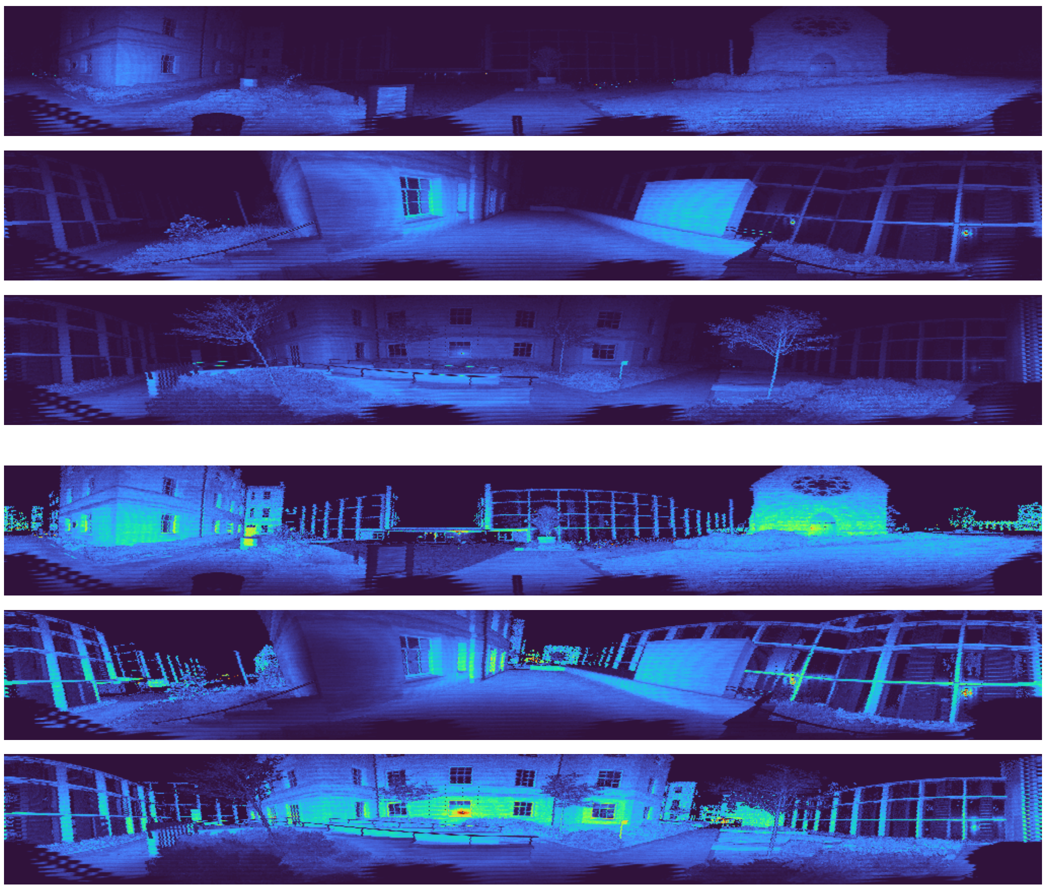
Figure 6.3: Intensity values to [0,0.35] (top) and reflectivity values (bottom) of the math-medium dataset

ration of the scene. Math-Medium increases speed and adds independent sensor rotation. Math-Hard increases both walking and rotational speed and adds rapid shaking.

Mine

The Mine dataset is not mentioned in the Newer College paper [L. Zhang, Camurri, and Fallon 2021] but is available from the same source. It contains LiDAR measurement of an underground mine. It consists of enclosed spaces and very close ranges. Most of the path follows straight tunnels which are interrupted by intersections.

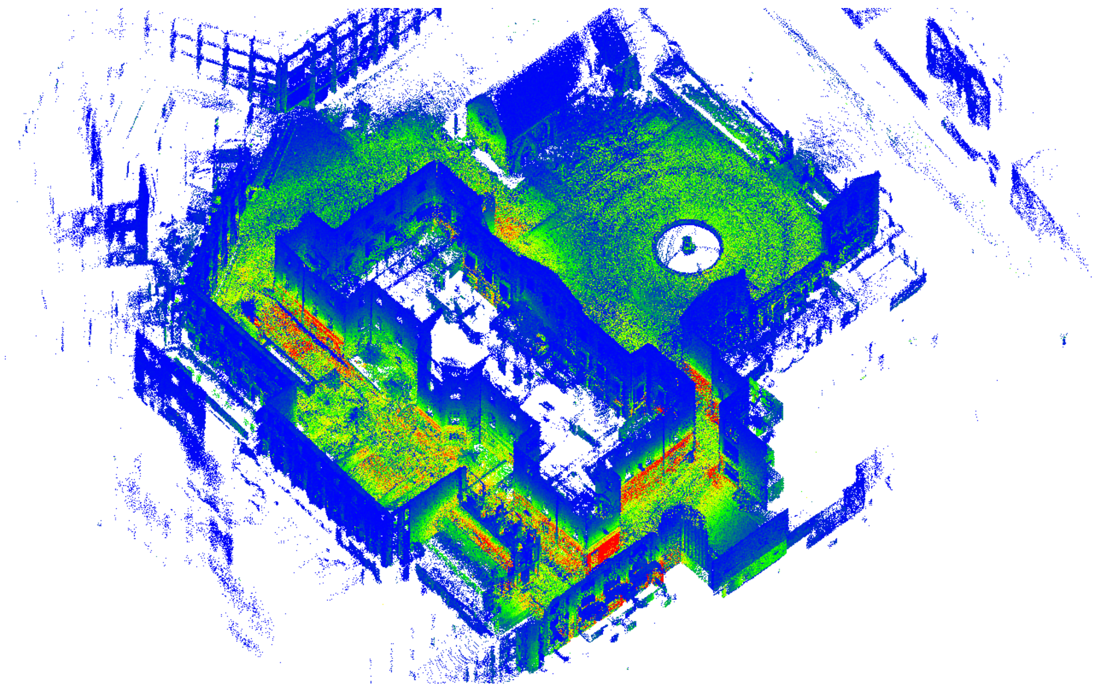

Figure 6.4: Map of the Math-Medium dataset colored by intensity

Sensor: Ouster OS0-128

The Ouster OS0-128 is a MOSFET-based LiDAR scanner operating using the combined beam-steering, flash LiDAR principle. The vertical opening angle is 90° with 360° rotation and 128 vertical rings. The sensor consists of 128 receiver units that sit offset in sets of 4. This means that during a flash each point recorded is slightly horizontally offset.

The sensor provides not only range and intensity information, but also reflectivity. This channel information uses the recorded range and intensity to precompensate intensity by range. In a second step the lower value intensity is scaled lineary, but higher values are scaled with $\log_2$. This allows for reflectivity to more accurately depict most points as intensity tends to clump towards the lower values. Fig. 6.7 shows the intensity distribution against range for each scene of the Newer College dataset.

6.2 Compensation Models

The models shown in Tab. 6.1 where trained on the Quad-Easy dataset fusing 12 point clouds with minimum pose difference of $1m$. They show some initial problems with parameter optimization. Point pairs where filtered aggressively

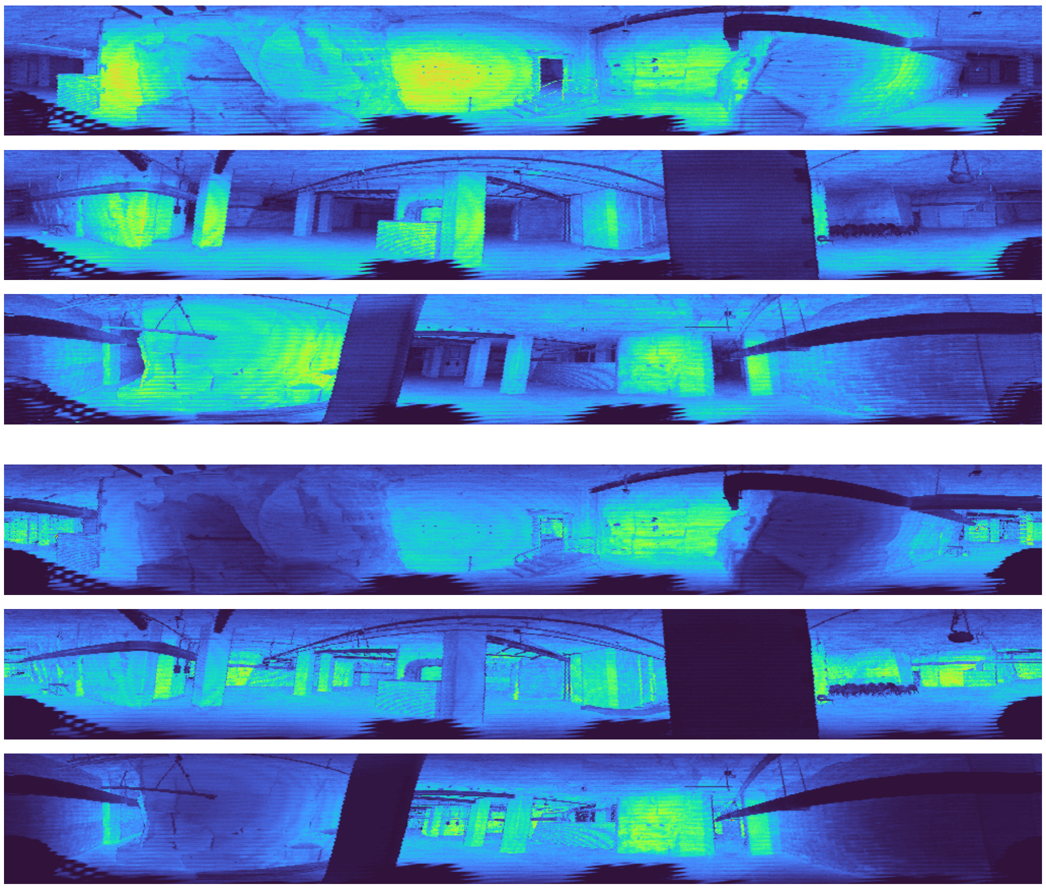

Figure 6.5: Intensity values to [0,0.35] (top) and reflectivity values (bottom) of the Mine dataset

w_r	w_α	r_{mid}	r_{min}	ϕ_{wave}	ϕ_{vign}
1.336	-0.0889	-	-	-	-
1.341	-0.0912	6.06982	0.353138	-	-
1.666	-0.0770	6.12945	0.523435	-	$-4.57, 9.267, -3.678$
1.341	-0.0913	6.07069	0.352381	$1.3, 0.20, -0.012$	-
1.4	-0.1123	4.95	0.5025	$2.54, 0.2, -0.026$	$3.88, -5.45, 4.63$

Table 6.1: Initial model parameters optimized using the Quad-Easy dataset. High w_r and low w_α can be seen. Models with just vignette but no wave compensation show higher absolue values in ϕ_{vign}

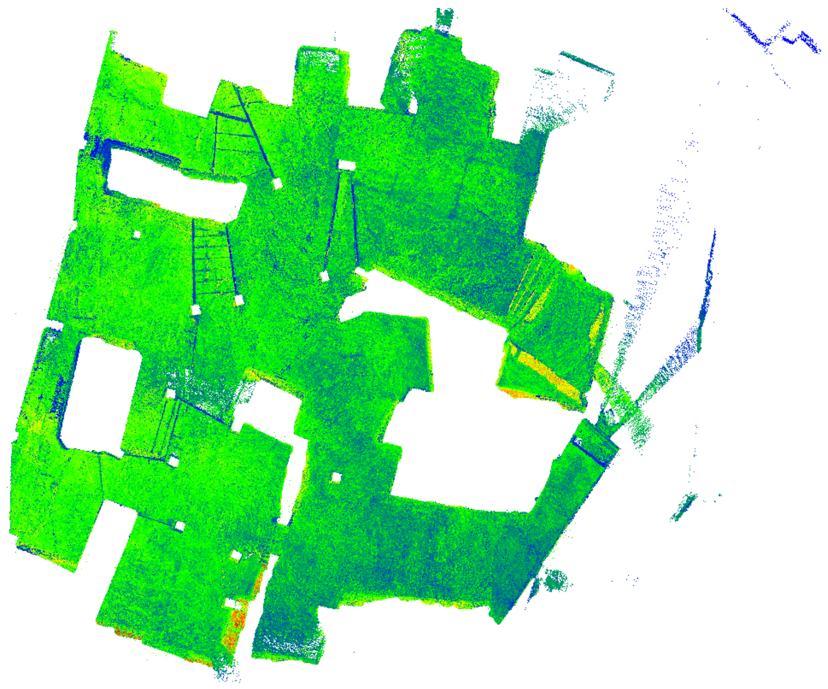

Figure 6.6: Map of the Mine-Medium dataset colored by intensity

demanding Eq. 4.18 and Eq. 4.19 hold simultaneously. What can be seen is that w_r is estimated relatively high, while w_α is comparatively small when compared to Tab. 6.5. Compensation generated by these models can be seen in Fig. 6.8. Objects further away are noticeable brighter and closer areas show non-even intensity. For models including c_{vign} difference in intensity get overcompensated, resulting in nearly equal intensity for all surfaces. Also the compensated intensity distribution shows the overestimation of w_r. This indicates that the estimated parameters do not compensate well.

Better results for w_r could be achieved using $\Delta_r = \Delta_\alpha = 0.05$ and only enforcing one of them must hold. A large number of points is taken inside the initial tunnel area. Testing in enclosed areas show that such areas estimate lower w_α. Enlarging the trajectory by using minimum pose distance of $2m$ increases the estimated w_α. This results in more even intensity.

Another problem for optimization is foliage and comparable areas with large localized angle variance. Fig. 6.10 shows the intensity and angle of a tree inside the fused point cloud. The resulting parameters are very different compared to other scenes. Tab. 6.2 shows how parameters change. The tree is a major part of scans 100 and 200. For scans 300 and 400 part of the tree and other bushes are part of the pointcloud. Scan 500 still has some bushes in the dataset resulting in values very different from other models.

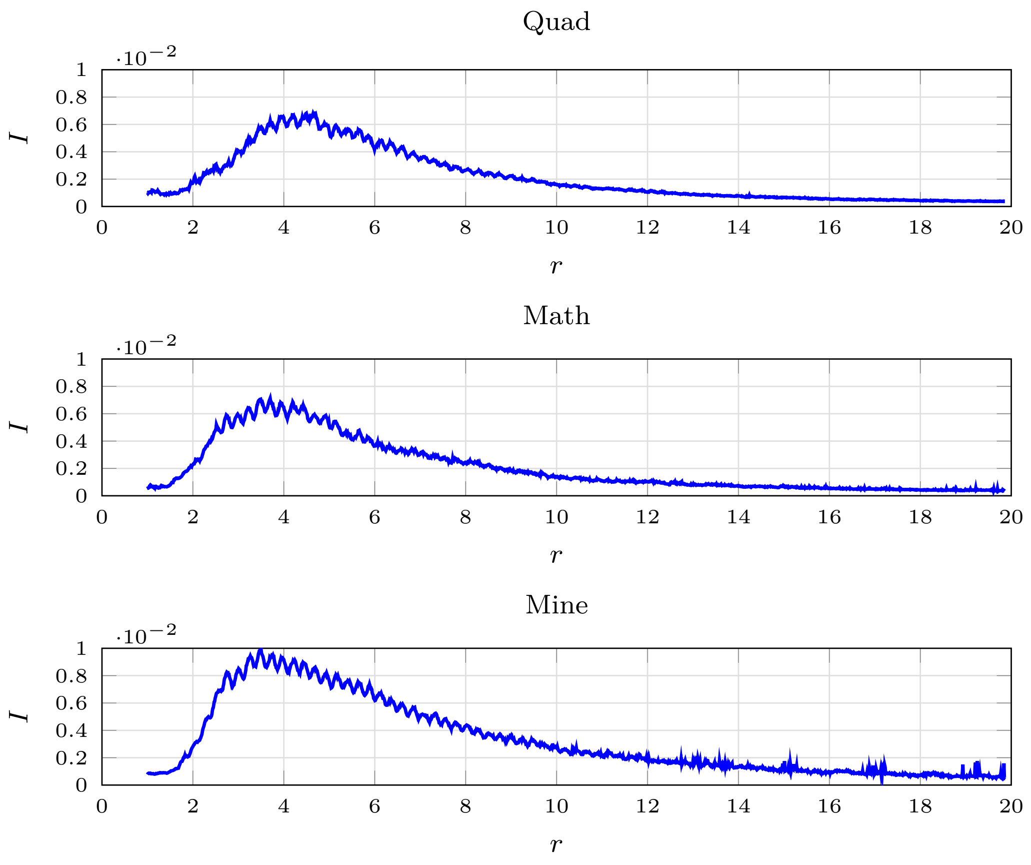

Figure 6.7: Intensity measurements plotted against range for Quad, Math and Mine of the Newer College dataset. For all scans intensity decreses for $r < 4$. Also a very visible wave-like noise can be seen on all datasets. Some dataset characteristics can be distinguished e. g. Mine shows higher intensity over all range steps.

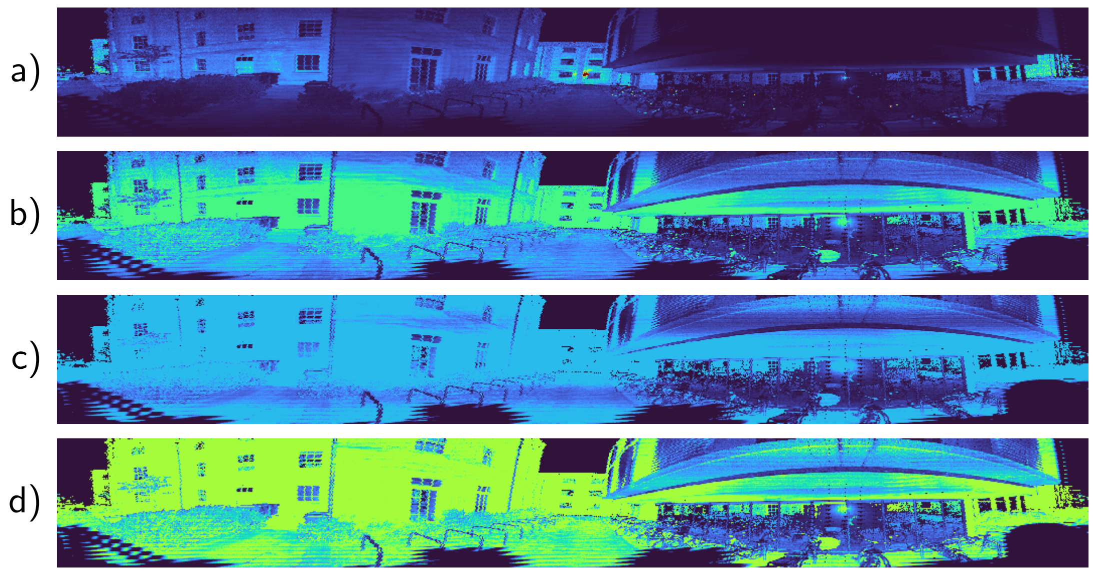

Figure 6.8: Compensated intensity images for the models shown in Tab. 6.1 row 1 (a), row 2 (b), row 3 (c) and row 5 d). Without vignette compensation close walls are not evenly compensated. c) shows the artifacts introduced by to large ϕ_{vign}. c) and d) also show the combination of large w_r. Different surfaces cannot be distinguished by intensity anymore.

Scan	w_r	w_α	r_{mid}	r_{min}	
0	0.8896	-0.2188	3.9945	0.9201	
100	2.8438	-0.0016	34.121	0.8229	
200	2.8369	0.0303	2×10^{10}	0.7546	
300	0.9683	-0.1257	3.957	0.9809	
400	0.7791	-0.2964	3.4851	1.1346	
500	1.1801	-0.3514	7.1438	0.5519	
600	1.2029	-0.166,	4.0915	1.2904	
700	1.1506	-0.1715	4.9701	0.941	
800	1.1616	-0.276	4.179	1.1537	
900	0.8422	-0.1341	4.1452	0.926	

Table 6.2: Model values for exponential weight and angle with cos4 optimized along multiple positions on the Quad-Easy dataset. Start scans 100 and 200 include a tree and other plants which results in large differences in parameters

Fig. 6.9 shows the quad area. The operator starts at a pose e. g. 200 and moves clockwise. A trajectory using 12 poses with minimum pose dist $2m$, as used for the models in Tab. 6.2, spans around 200 poses. As can be seen starting at 0 starts in the entry tunnel and stops right before pose 200. While some foliage information is part of the fused point cloud their is a lot of other information that constrains the optimization. Starting from pose 600 no more bushes or trees are part of the sampled fused point cloud. This results in a noticeable drop in w_α. Most differences in angle are only observed along the flat walls and the ground.

In general wide open areas e. g. the lower side of quad in Fig. 6.9 result in larger estimations of w_r. This might also be because without a sufficient amount of variance in α the effect of α is usually underestimated. For good parameter optimization a difference in r e. g. moving towards a wall, combined with observing closer objects, for which a usable α estimation can be done from different angles, such as moving in a semi circle along a wall. This keeps r somewhat constant and improves estimating w_α.

Optimizing reflectivity models is more difficult. The non-linear scaling can lead to bad parameters, see e. g. model **rwpC** Tab. 6.4. While most remarks regarding intensity models tend to transfer to reflectivity optimization, high intensity points like windows and other reflecting surfaces should be avoided.

Optimizing c_{vign} without wave compensation is difficult. As can be seen in Fig. 6.8 c) larger estimation of ϕ_{vign} show visible artifacts in an area along the upper and lower area of the images. Optimization of c_{vign} can be fragile with respect to input point cloud. No changes to optimization parameterization tried, solved this issue consistently over all valid input point clouds. It is therefore advised to not use c_{vign} without also using c_{wave}.

Comparing compensation models in Fig. 6.11 shows how different parameters change intensity. High $w_r > 1.0$ results in visible overcompensation of far away objects. This is especially visible for ground patches. The compensation found for models a) and c) create more even intensity values of far walls compared to close walls. Also ground areas look more even. Models a) and c) show visible overcompensation of very close objects. As they are trained on Quad-Easy comparatively few points at $r < 2m$ were involved in model optimization.

All models do not compensate α sufficiently. As w_r seems to be well estimated in a) and c) but walls still contain a visible horizontal gradient. To improve this point clouds used for optimization might need to be better aligned and movement compensated. Some last minute tests suggest that a better value for w_α might be around -0.4.

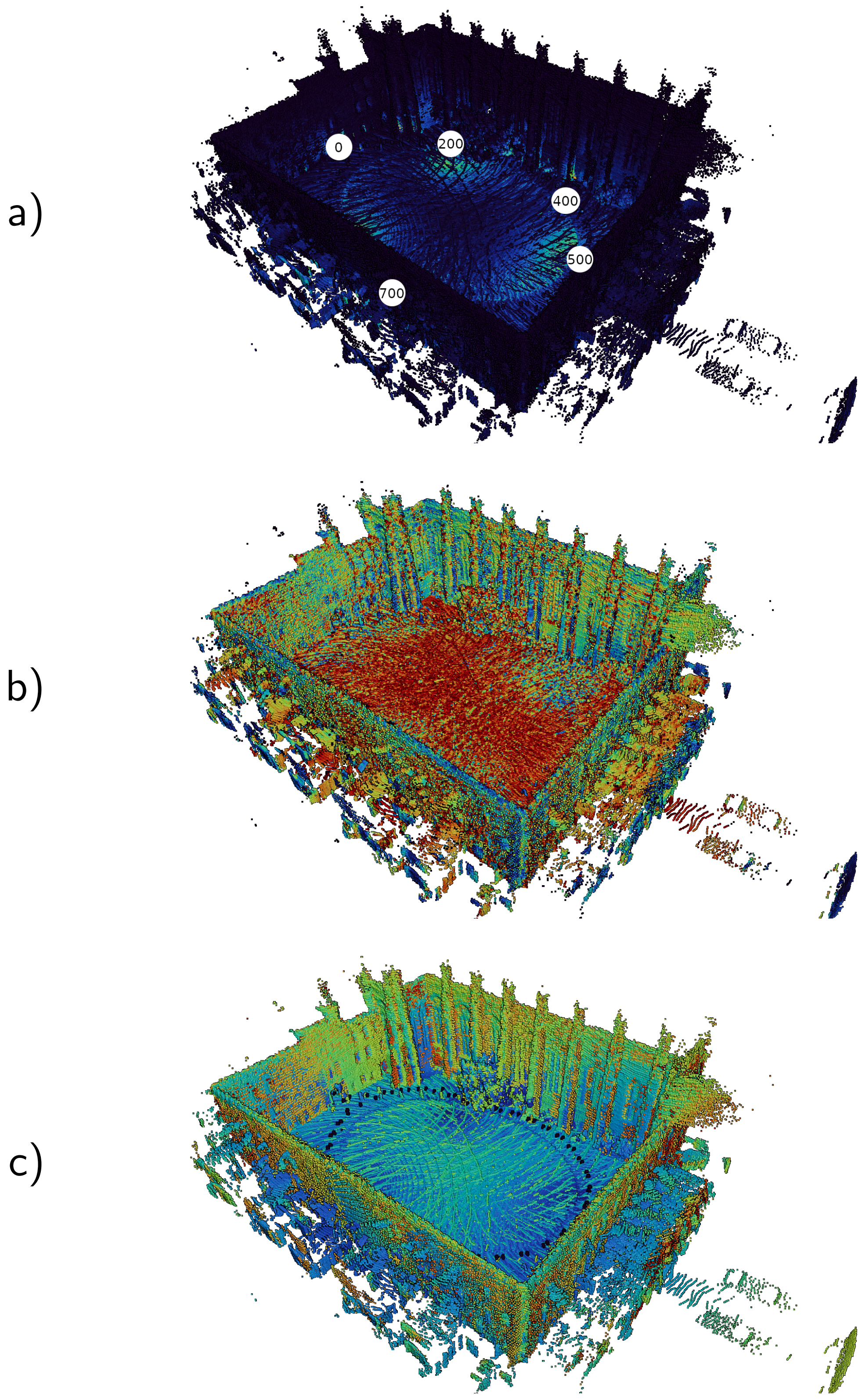

Figure 6.9: Fused point cloud generated from Quad-Easy, using 40 point clouds starting at scan 0, with minimum pose distance of 3m. a) shows intensity as well as the start positions for some of the models in Tab. 6.2. The number references the n-th pose. b) shows the estimated angles and c) the measured distance of the dataset.

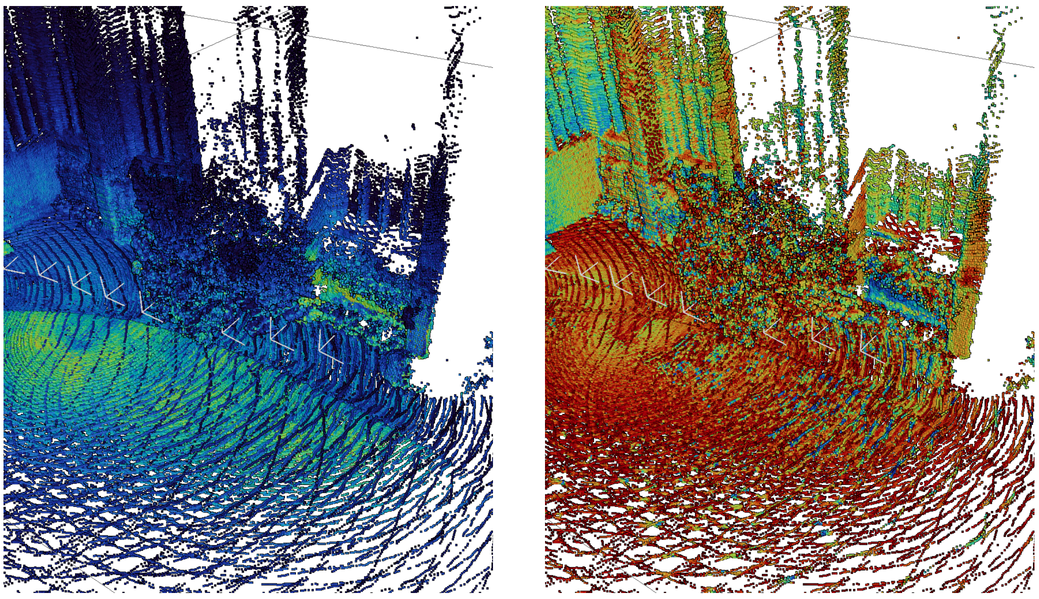
Figure 6.10: Points sampled from Quad-Easy for optimization when starting at pose 120. Shown is intensity (left) and estimated angles (right). As can be seen while intensity is even, angles vary widely. This prevents the optimization from generating usable parameters.

6.3 Localization Performance

Evaluation Setup

Experiments were conducted on an Intel NUC Bean Canyon 8i7BEH with a Core i7-8559U processor and 32Gb of RAM. The data set used is the Newer College dataset described in Sec. 6.1.

The metrics used to compare the localization performance of updated FastLIO2 are Average-Trajectory-Error (ATE) and Relative-Pose-Error (RPE). These are computed using the evo package [Grupp 2017] for python, which also computes the trajectory alignment.

As trajectories may have arbitrary initial poses, they have to be aligned before comparison. Given trajectory $\{\boldsymbol{T}_1, \dots, \boldsymbol{T}_n\}$ and corresponding ground-truth trajectory $\{\hat{\boldsymbol{T}}_1, \dots, \hat{\boldsymbol{T}}_n\}$, the is computed using the Umeyama alignment method [Umeyama 1991]. This results in rotation $\hat{\mathbf{R}}$ and translation $\hat{\boldsymbol{t}}$. The ATE can now

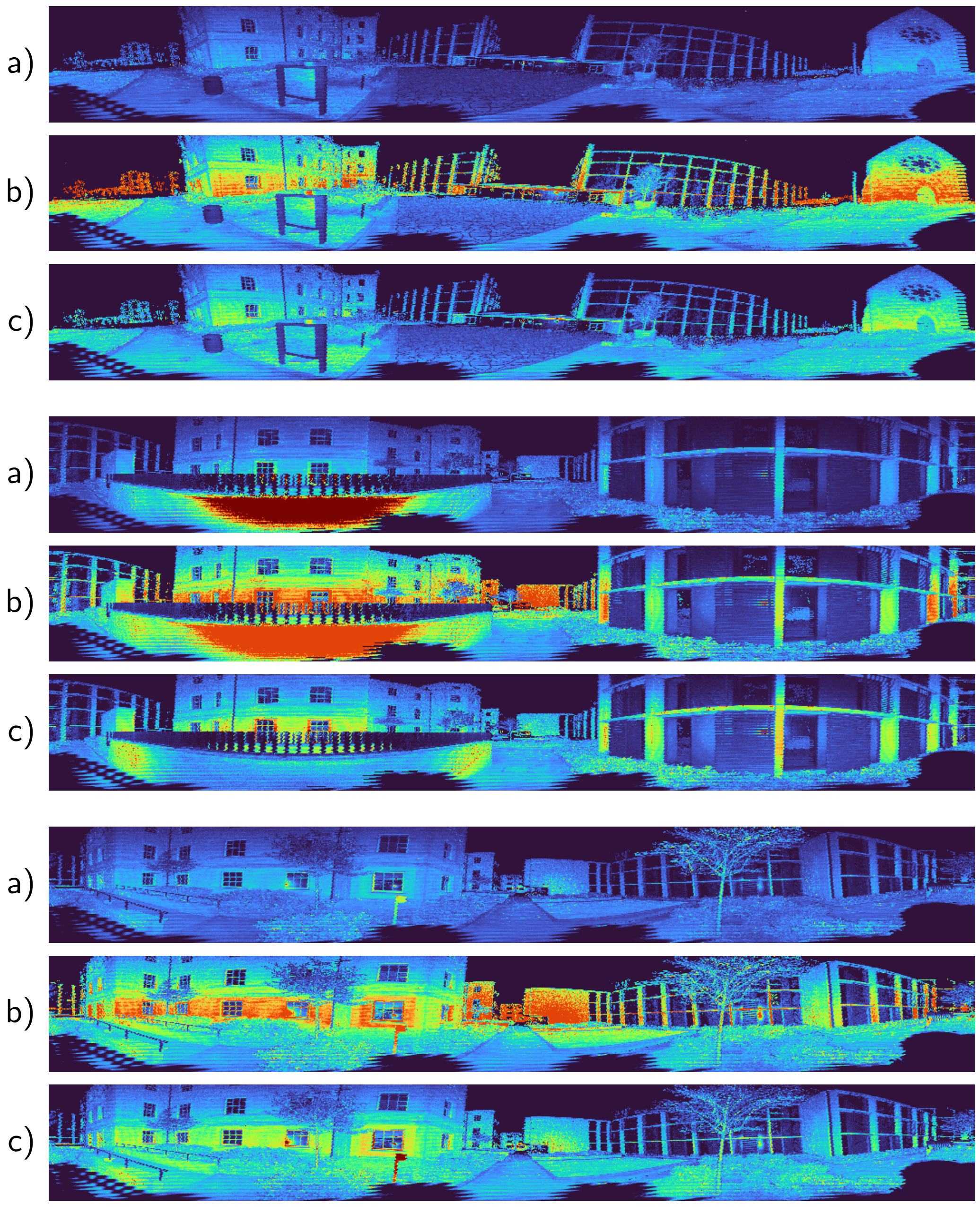

Figure 6.11: Multiple projected point clouds taken from Math-Medium. In each triplet a) is created using model in Tab. 6.2 0, b) uses 600 and is rescaled by factor 0.83 to reduce saturation and c) uses model c42 in Tab. 6.3 and is rescaled by factor 2.

be defined as:

$$\mathbf{E}_i^a = \hat{\boldsymbol{T}}_i^{-1} \begin{pmatrix} \hat{\mathbf{R}} & \hat{\boldsymbol{t}} \\ 0 & 1 \end{pmatrix} \boldsymbol{T}_i = \begin{pmatrix} \hat{\boldsymbol{R}}_i^a & \hat{\boldsymbol{t}}_i^a \\ 0 & 1 \end{pmatrix}, \tag{6.1}$$

$$\text{ATE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} ||\hat{\boldsymbol{t}}_i^a||^2}. \tag{6.2}$$

This is the root-mean-squared error (rmse) of the ATE.

Given a fixed time interval Δ_t RPE is defined as:

$$\mathbf{E}_i^r = \left(\hat{\boldsymbol{T}}_i^{-1} \hat{\boldsymbol{T}}_{i+\Delta_t}\right)^{-1} \left(\boldsymbol{T}_i^{-1} \boldsymbol{T}_{i+\Delta_t}\right) = \begin{pmatrix} \hat{\boldsymbol{R}}_i^r & \hat{\boldsymbol{t}}_i^r \\ 0 & 1 \end{pmatrix}, \tag{6.3}$$

$$\text{RPE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} ||\hat{\boldsymbol{t}}_i^r||^2}. \tag{6.4}$$

This error measures the difference between pose differences. Poses are matched by timestamp. Evo allows to set a manual time offset, if the estimated trajectory is offset, due to e. g. computation time.

Tested Models

The tested baselines are: FastLIO2, FastLIO2 with point filter and Raw Reflectivity. An overview of tested models can be found in Tab. 6.3 for intensity models and Tab. 6.4 for reflectivity models. The models except rwpC where optimized using the same fused point cloud using 12 poses with minimum pose difference of 2m but with different parameters for Δ_r and Δ_α. The fused point cloud can be seen in Fig. 6.12. The different training parameters are referenced as set 1 and 2.

Set 1 contains **wra1** which only used w_r and w_α, **c41** which also uses close range cos4, **p1** which uses c4 and adds c_{vign},**w1** which instead adds c_{wave} and **wp1** which uses all these. Further reflectivity models **rwra1** which again only uses w_r and w_α and **rwp1** which again uses all compensation parts. Set 2 uses the same keys as set 1 **w2** e. g. stands for w_r, w_alpha with cos4 and c_{wave}. Another reflectivity model **rwpC** is added. This model did not converge during optimization and the parameters are very different to any comparable model. This was kept as control.

This results in 9 intensity models and 4 reflectivity models as not all model types are present in each set. Each model was tested on all Newer College sequences, except Stairs, using different values for w_I.

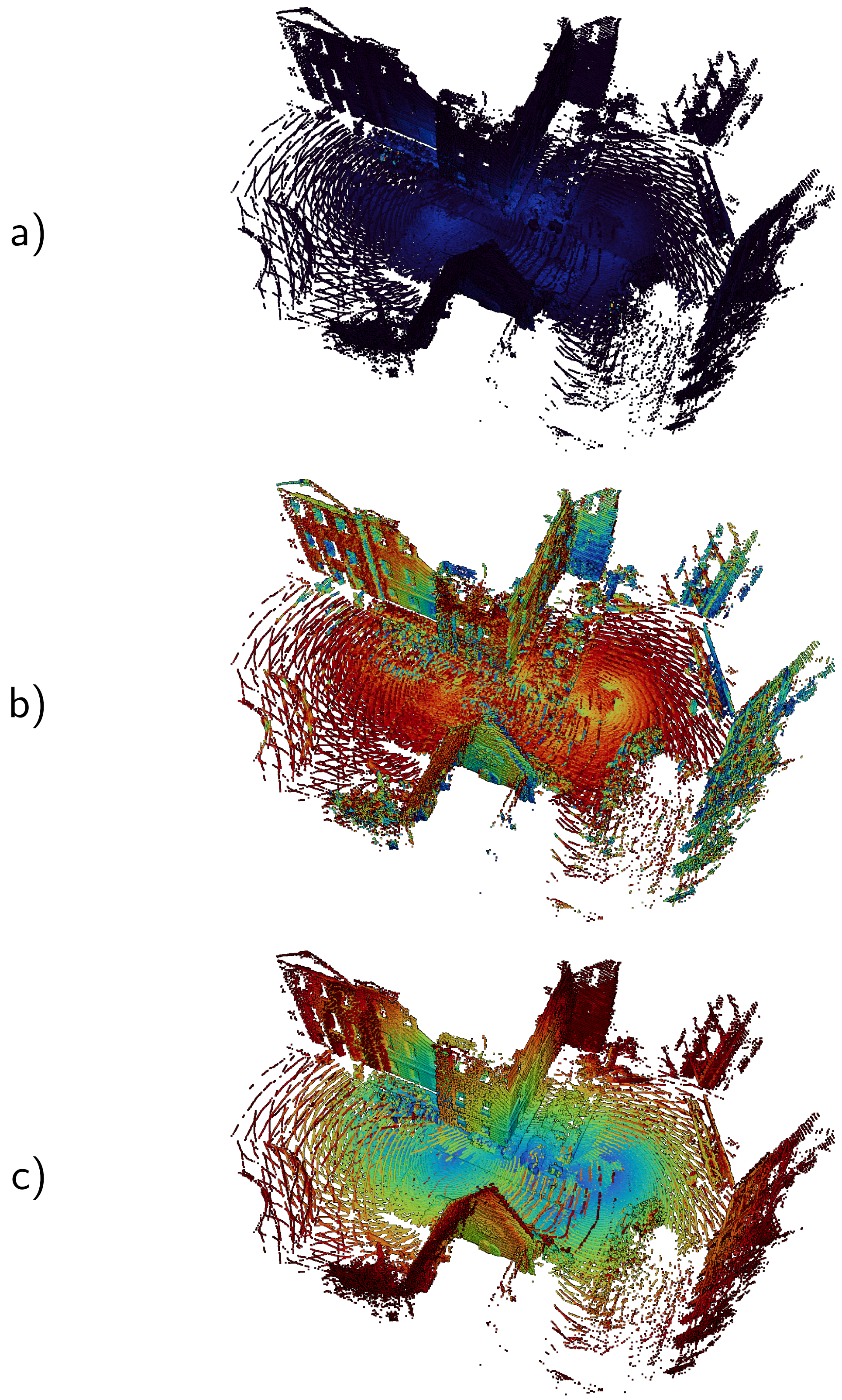

Figure 6.12: The fused point cloud used to train most models used to test localization perfomance. The used point clouds contains high variance in intensity a), a good variation in estimated α b) and a combination of near and far ranges.

Key	w_r	w_a	r_{mid}	r_{min}	ϕ_{wave}	ϕ_v
wra1	0.761	-0.131	-	-	-	-
c41	0.770	-0.128	6.12	-1.27	-	-
c42	0.896	-0.152	5.699	0.078	-	-
p1	0.909	-0.132	5.48	-0.52	-	$-3.99, 7.484, -3.947$
p2	0.949	-0.161	5.89	-0.069	-	$3.13, -7.45, 6.08$
w1	0.787	-0.124	6.512	-1.493	$5.0, 0.20, -0.079$	-
w2	0.884	-0.166	5.635	0.077	$-153.8, 0.18, 0.091$	-
wp1	1.189	-0.190	4.681	0.844	$9.96, 0.2, -0.111$	$0.163, -2.81, 3.38$
wp2	0.968	-0.141	5.717	0.026	$-150.9, 0.185, -0.092$	$3.14, -7.33, 5.96$

Table 6.3: Intensity models used for evaluating localization performance. All models were trained on Math Medium using the same starting position.

Key	w_r	w_a	r_{mid}	r_{min}	ϕ_{wave}	ϕ_v
rwra1	-0.339	-0.141	-	-	-	-
rw2	-0.286	-0.237	5.636	0.077	$-153.8, 0.18, 0.09$	-
rwp1	9e-4	-0.199	2.245	1.272	$0.8, 0.2, -0.1$	$6.54, -9.97, 5.87$
rwpC	2.362	-0.04	25.54	0.991	$-4.14, 0.23, -2.39$	$-14.1, -55.4, -55.4$

Table 6.4: Reflectivity models used for evaluating localization performance. All models were trained on math medium using the same starting position. Model rwpC was trained on a different dataset and did not optimize correctly. It was kept as control.

	time offset 0.0			time offset 0.1		
Model	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
FastLIO2	0.1170	0.1460	0.1688	**0.0793**	**0.1080**	0.0680
Filtered Points	0.1083	0.1387	0.1343	0.1488	0.1582	0.0929
Raw Ref	0.1054	0.132	0.1265	0.1341	0.1562	0.0920
wra1	0.1042	0.1291	0.1522	0.0834	0.1131	0.0662
c41	0.1035	0.1297	0.1373	0.0831	0.1167	0.0664
c42	0.1000	0.1285	0.1514	0.0834	0.1137	0.0665
p1	0.1038	0.1297	0.1522	0.0836	0.1131	0.0673
p2	0.1002	0.1288	0.1373	0.0831	0.1123	0.0664
w1	0.1044	0.1297	0.1521	0.0824	0.1129	0.0664
w2	0.0998	0.1289	**0.1026**	0.0832	0.1124	0.0658
wp1	0.1047	0.1297	0.1525	0.0835	0.1141	**0.0651**
wp2	0.1006	0.1288	0.1512	0.0836	0.1131	0.0718
rwra1	0.0990	0.1310	0.1509	0.1351	0.1624	0.1018
rw2	0.0994	0.1315	0.1320	0.1346	0.1627	0.1012
rwp1	X	X	X	X	X	X
rwpC	**0.0921**	**0.1238**	0.14	0.1289	0.1588	0.0965

Table 6.5: ATE for the three sequences of the Math-Institute dataset of Newer College. X marks non-convergence. Bold numbers are the best on sequence. Time offset refers to the time offset used to compare trajectories. An offset of 0.1 means that an estimated pose at time t is compared to pose $t + 1$ in the reference trajectory.

6.3.1 Performance on Newer College

Math Institute

Tab. 6.5 shows the ATE for all models on the Math-Institute dataset. For all intensity models $w_I \in \{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}\}$ where tested and the best results are displayed. $w_I = 1 \times 10^{-4}$ seemed to work best. Small tests with $w_I < 1 \times 10^{-4}$ suggest that the best weight for intensity on the math set seems to be around 1×10^{-4}. This transfers for all other datasets. While using slightly smaller or larger weights tended to increase performance slightly. finding the right w_I for each model and datasets is very time consuming.

Noticeable is that rwp1 diverged while rwpC generates the best results for Math-Easy and Math-Medium. Differences between set 1 and 2 seem consistent on Math-Easy and medium, with set 2 models being slightly better in all comparisons. Performance on Math-Hard seems to be very sensitive to chosen w_I dependent on model. Models c41, p2 and especially w2 seem to work better with chosen w_I.

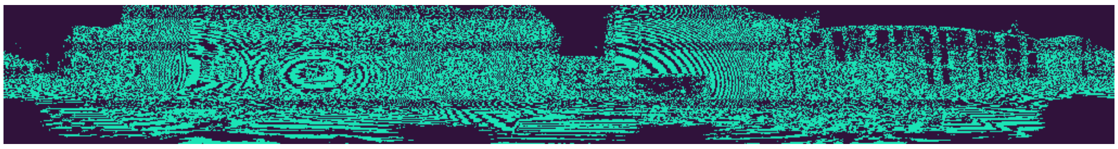

Figure 6.13: Model rwpC creates a wave pattern from given reflectivity input cloud. Also visible are c_{vign} artifacts. As the pattern is not stable with position but instead moves depending on range, it should not serve as usable features for localization.

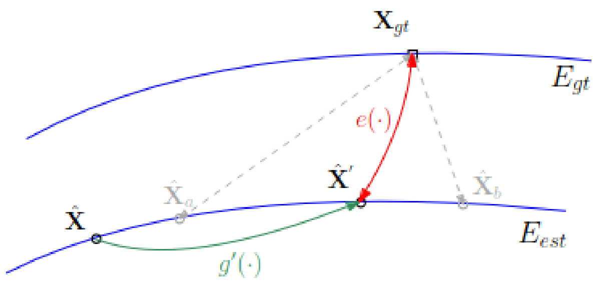

Figure 6.14: Figure taken from Z. Zhang and Scaramuzza 2018 Fig. 5. Despite similar trajectories comparing estimated pose $\boldsymbol{X}_a$ or $\boldsymbol{X}_b$ to given $\boldsymbol{X}_{gt}$ can lead to large differences in APE. Although the trajectories do not change, just adding a pose offset, switching $\boldsymbol{X}_a$ for $\boldsymbol{X}_b$ can drastically change the resulting score.

The model rwpC generates no usable reflectivity compensation, an example is shown in Fig. 6.13. It is therefore surprising to find it doing so well for time offset 0. For time offset 0.1 the model is much more in line with expectations. According to Zhang et al. [L. Zhang, Camurri, and Fallon 2021] the ground truth poses are generated by matching the undistorted point clouds, as are used by FastLIO to high quality pre registered maps. Hence time offset 0.1 should be incorrect. Also looking at the RPE suggests that time offset 0 is correct, as RPE on average is lower for time offset 0, see Tab. 6.6. Fig. 6.14 shows a possible reason for this odd behavior.

A comparison of a generated map of Math-Medium without and with intensity compensation can be seen in Fig. 6.15. The compensated maps look more even and different surfaces can be more easily distinguished. The horizontal gradient present also in the 2d projections can be seen in the map also.

Quad

Tab. 6.7 shows the ATE for the Quad datasets. Again model set 2 performs slightly better than set 1. Differences for Quad-easy and Quad-medium are minimal.

Dataset	time offset 0.0	time offset 0.1
Math-Easy	0.0609	**0.05927**
Math-Medium	0.12623	**0.12526**
Math-Hard	**0.05242**	0.0543
Quad-Easy	**0.03738**	0.03857
Quad-Medium	**0.12541**	0.126
Quad-Hard	**0.09547**	0.0967
Mine-Easy	**0.03701**	0.03796
Mine-Medium	**0.06765**	0.06847
Mine-Hard	**0.10781**	0.10978
Park	**0.05644**	0.05948
Cloister	**0.04516**	0.04764

Table 6.6: RPE for the **w2** model for time offset 0.0 and 0.1. As can be seen the RPE for most datasets is lower for offset 0.0. This holds also for most models.

	time offset 0.0			time offset 0.1		
Model	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
FastLIO2	0.0924	0.0996	0.1471	0.0693	**0.0611**	0.0566
Filtered Points	0.0894	0.0952	0.1564	0.0889	0.0859	0.1139
Raw Ref	0.0892	0.0950	**0.1372**	0.0887	0.0855	0.0646
wra1	0.0883	0.0899	0.1438	**0.0691**	0.0618	0.05778
c41	0.0887	0.0895	0.1474	0.0706	0.0638	0.0589
c42	0.0883	0.0891	0.146	0.0703	0.0635	0.0605
p1	0.0887	0.0894	0.1490	0.070	0.0626	0.0575
p2	0.0882	0.0886	0.1474	0.0706	0.0638	0.0589
w1	0.0885	0.0891	0.1501	0.0702	0.0619	0.0583
w2	**0.0722**	0.0889	0.1481	0.0699	0.628	0.577
wp1	0.0884	**0.0883**	0.1463	0.0707	0.0624	**0.0557**
wp2	0.0884	0.0889	0.1467	0.0702	0.0627	0.0575
rwra1	0.0964	0.1019	0.1392	0.0988	0.0974	0.0641
rw2	0.0961	0.1023	0.1458	0.0984	0.0971	0.0689
rwp1	X	X	X	X	X	X
rwpC	0.0903	0.0991	0.1543	0.0833	0.0951	0.0789

Table 6.7: ATE for the three sequences of the Quad dataset of Newer College. X marks non-convergence. Bold numbers are the best on sequence.

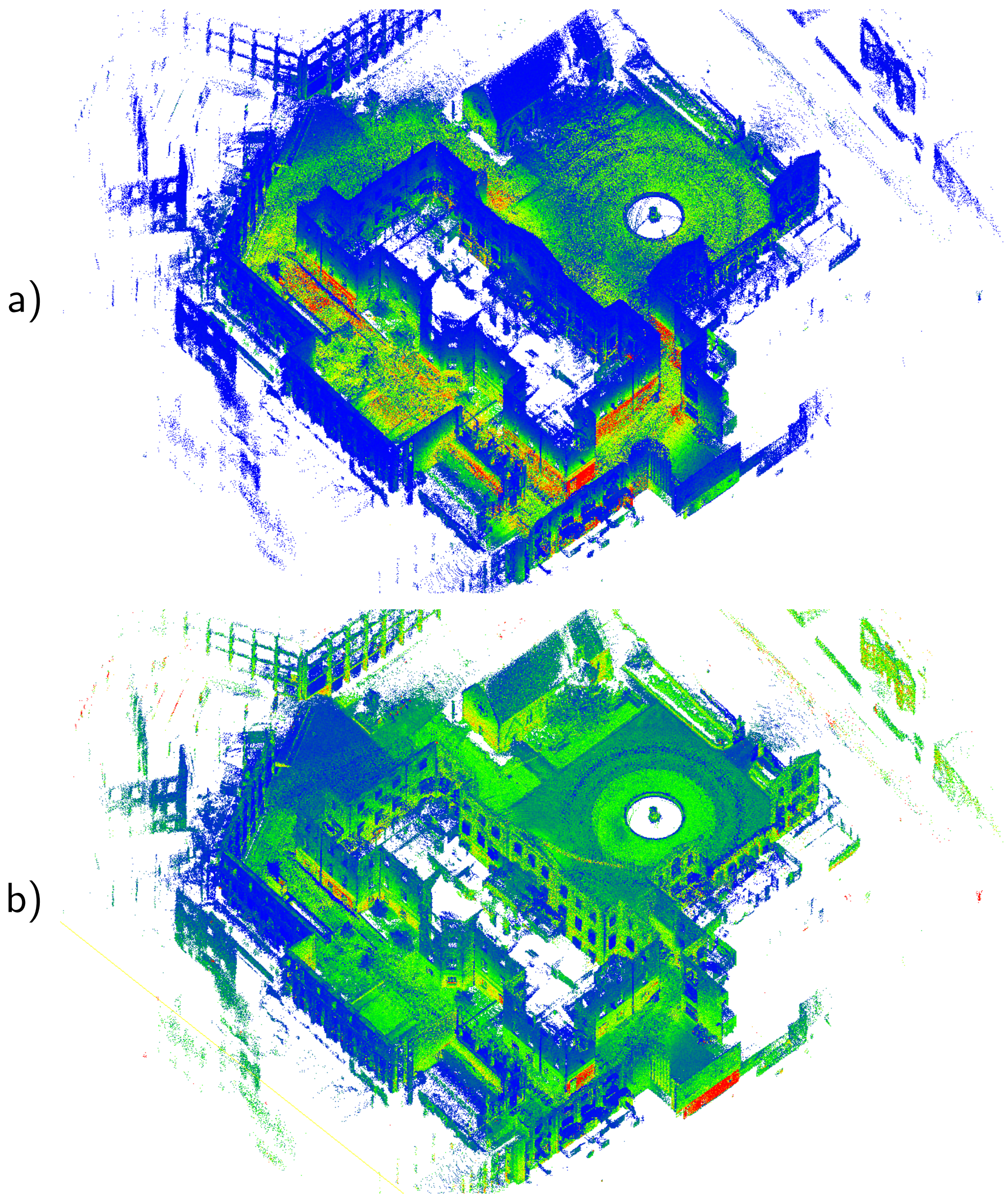

Figure 6.15: The map of Math-Medium generated by FastLIO without using intensity compensation a) and with compensated intensity b). Ground areas show more even intensity in the compensated map. Elements like walls, curbs and plants can be discerned.

	time offset 0.0			time offset 0.1		
Model	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
FastLIO2	0.1558	0.1533	0.1417	0.0496	0.0462	0.0525
Filtered Points	0.1188	0.1343	0.1318	0.0474	0.0434	0.0656
Raw Ref	0.1152	0.1265	0.1244	0.0416	**0.0371**	**0.0544**
wra1	0.1317	0.1343	0.1261	0.0357	0.0377	0.0561
c41	0.1317	0.1347	0.1266	0.0364	0.0376	0.0567
c42	0.1290	0.1365	0.1305	0.0358	0.0375	0.0563
p1	0.1342	0.1374	0.1304	0.0359	0.0372	0.0560
p2	0.1290	0.1366	0.1304	0.0357	0.0377	0.0568
w1	0.1322	0.1368	0.1303	0.0379	0.0376	0.0550
w2	0.1292	0.1364	0.1299	0.0377	0.0379	0.0568
wp1	0.1336	0.1371	0.1305	**0.0355**	0.0376	0.0576
wp2	0.1292	0.1366	0.1303	0.0359	0.0377	0.0562
rwra1	0.1147	0.1273	0.1244	0.0413	0.0375	0.0547
rw2	0.1292	**0.1267**	0.1244	0.0357	0.0376	0.0550
rwp1	X	X	X	X	X	X
rwpC	**0.1125**	0.1282	**0.1238**	0.0419	0.0416	0.0578

Table 6.8: ATE for the three sequences of the Mine dataset of Newer College. X marks non-convergence. Bold numbers are the best on sequence.

Reflectivity models again perform worse than baseline uncompensated reflectivity. A map generated with compensated intensity can be seen in Fig. 6.16 (top). The inner grass area can be easily discerned from surrounding pavement. Differences between walls and pavement are less obvious.

Mine

Tab. 6.8 shows the results from the Mine dataset. The Mine dataset contains exclusively enclosed spaces. It is also the dataset with the greatest difference between time offset 0.0 and 0.1. Reflectivity is competitive compared to intensity. Fig. 6.16 (bottom) shows a map with compensated intensity. There is little difference between that map and the uncompensated map in Fig. 6.6. Especially compared to the difference seen between other maps.

Park and Cloister

The results for the park and cloister datasets can be seen in Tab. 6.9. Again reflectivity performs well on the enclosed spaces of cloister, while compensated intensity performs worse than baseline.

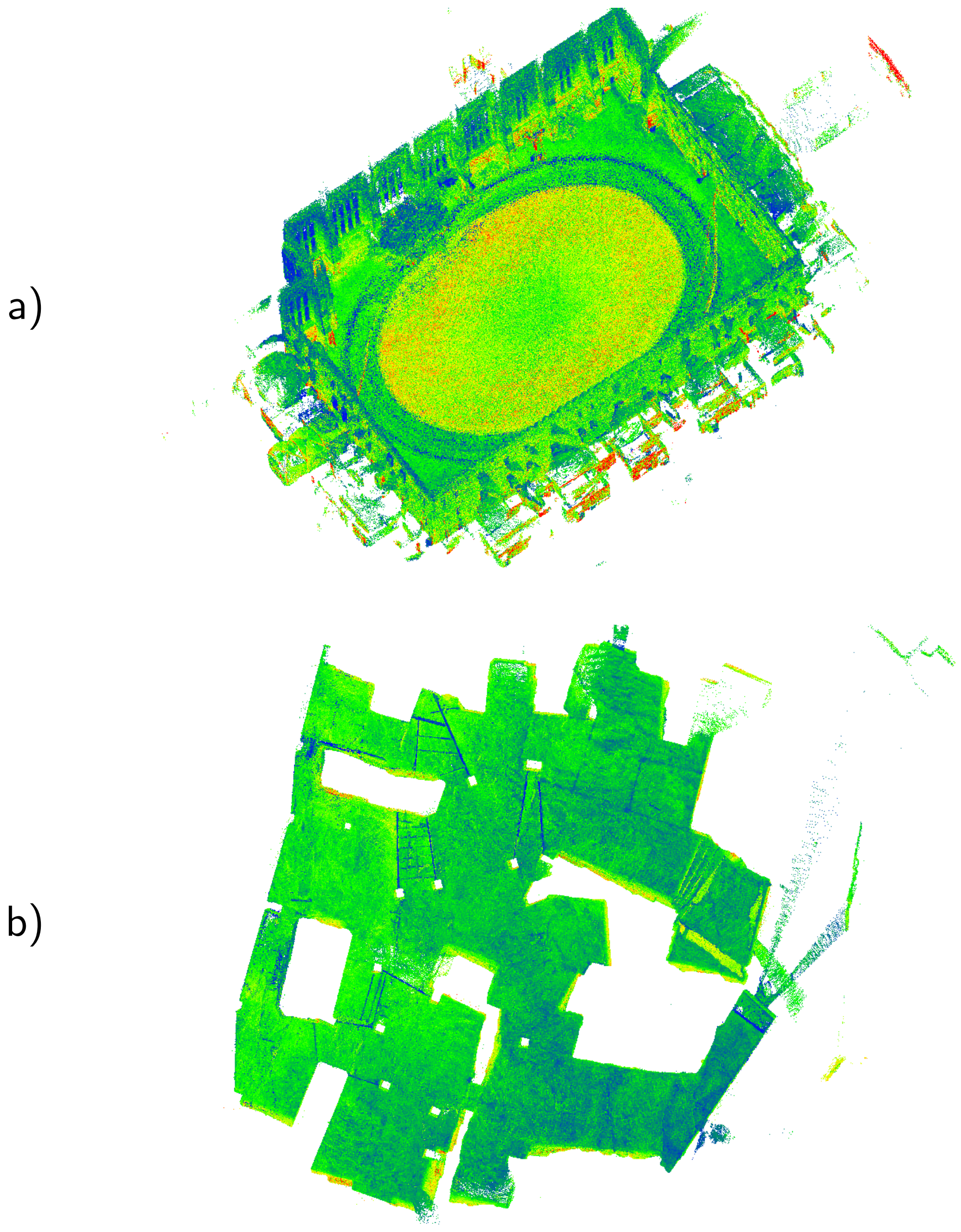

Figure 6.16: The map of Quad-Easy a) and Mine-Medium b) with compensated intensity. Quad-Easy shows a noticeable difference to the inital uncompensated intensity. The center oval has even intensity and can be clearly differentiated from surrounding areas. The compensated intensity does not change the Mine-Medium map much, as the initial map already contained a large amount of high intensity points.

	time offset 0.0		time offset 0.1	
Model	**Park**	**Cloister**	**Park**	**Cloister**
FastLIO2	0.3667	0.1426	0.3181	0.0667
Filtered Points	X	0.1765	X	0.1094
Raw Ref	0.6958	0.1282	0.3102	0.0708
wra1	1.2706	0.1904	1.2449	0.1925
c41	0.3479	0.2045	0.3212	0.2005
c42	0.3486	0.2027	0.3234	0.2105
p1	0.3693	0.2029	0.3421	0.2102
p2	0.3492	0.1961	0.3198	0.2005
w1	0.3489	0.1888	0.3278	0.1849
w2	**0.3414**	0.1854	**0.3090**	0.1963
wp1	0.3443	0.1862	0.3172	0.1887
wp2	0.3466	0.2029	0.3166	0.2102
rwra1	0.5460	**0.1272**	0.5499	**0.0642**
rw2	0.6530	0.1284	0.6534	0.0666
rwp1	X	X	X	X
rwpC	0.6171	0.1436	0.6172	0.0826

Table 6.9: ATE for the cloister and park sequences of Newer College. X marks non-convergence. Bold numbers are the best on sequence.

Convergence on park tends to be more difficult. While using good values for w_I, usually 1×10^{-4} or 1×10^{-5} will improve results over baseline, some models do not converge at all. This is probably due to the large amount of trees in the second half of the dataset. While most datasets loose on average 50% of the input points due to filtering, park looses on average over 60% with areas loosing over 70%.

General Remarks

As can be seen compensated intensity can improve localization performance, but this depends on data context. In general reflectivity performs better in enclosed spaces and shorter range scenarios. This might be due to the difficulty during optimization leading to under performing reflectivity models.

Differences between intensity models tend to be limited. While parameterization may lead to some improvement in performance, see e. g. Tab. 6.5, differences between model types is limited.

Models using near range cos4 compensation and c_{wave} tend to have a slight edge on average. As these models are relatively easy to optimize, as the wave component can reliably be found. The remaining parameters are usually similar to the same model without c_{wave}.

Finding the perfect w_I currently means, searching for each model and each dataset a separate parameter. This is due to insufficient model normalization.

Performance

Given input point clouds with 131072 points, the proposed algorithm is capable of real-time compensation and localization. On the test system the compensation node on average needed 80-89ms per point cloud, while FastLIO2 needed 42-75ms depending on the amount of points passing the filter stage. This allows to process point clouds at a rate of 10Hz without issue.

7 Conclusion

This works presents a novel approach for compensating short range LiDAR sensors. The approach combines known physical concepts with observation based compensation for sensor effects.

The models can generate good parameter estimation using a small number of scans, that are sufficiently far apart to generate some variation in distance and angle. While the input scans greatly affect the model parameters some guidelines for good input scans could be presented. While the approach works well for intensity. Some more work regarding reflectivity is needed.

Applying these models to real world data shows that intensity can be visibly improved. Walls and comparable areas show more even intensity. Especially sensor specific problems like low intensity close range areas show improvements. The presented model for weighted range and angle with close range cos4 works well. While c_{wave} shows good results, c_{vign} introduces some problems regarding optimizing other parameters e. g. w_{α}. Problems c_{vign} tried to address are probably due to underestimation of w_{α}. Some more research into using rigid point cloud undistortion before fusing has offered some preliminary improvements.

In the second step of this work intensity was added to FastLIO2 to improve localization. The chosen approach takes the input point clouds, estimates local normals to retrieve α and uses this to compensate point clouds during FastLIOs runtime. The approach is real time capable.

First points are filtered by excluding outlier points and points, for which no normals can be estimated. This reduces the point cloud by a significant amount. As this mostly filters lone and outlier points this step alone improves FastLIO performance on a variety of datasets.

Intensity residuals are generated by estimating a local intensity gradient between map points. This is done by using the plane estimated by FastLIO2 for geometric residuals. The gradient on the plane is estimated and combine with the error between local intensity estimate and actual input intensity. This residual is added to the state estimation.

Tests on the Newer College dataset show improved localization performance using compensated intensity. Model type showed little changes in performance, parameter changes while also not large, influenced localization performance more.

Using compensated reflectivity is more difficult. While simple models without c_{vign} performed comparable to intensity model, models using vignette compensation diverged even for low values for w_I. Despite very small reflectivity gradients the trajectory tended to diverge fast.

The resulting frameworks proved to increase localization performance and generated maps showing more consistent pseudo-reflectance values.

As mentioned before to improve model optimization, applying rigorous point cloud undistortion to the used point clouds can be explored. Also looking more into point cloud statistics to better understand the relationship between optimized parameters and used points can be of interest.

To further improve localization performance, improving models might help, as better parameter estimation showed to improve performance. Also a more rigorous scheme for estimating local normals during intensity compensation for FastLIO can be explored.

List of Figures

List of Tables

Bibliography

Agarwal, Sameer, Keir Mierle, and The Ceres Solver Team (Mar. 2022). *Ceres Solver*. Version 2.1. URL: `https://github.com/ceres-solver/ceres-solver`.

Barsan, Ioan Andrei, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun (2020). "Learning to localize using a lidar intensity map." In: *Arxiv preprint arxiv:2012.10902*.

Bentley, Jon Louis (1975). "Multidimensional binary search trees used for associative searching." In: *Communications of the acm* 18.9, pp. 509–517.

Blanco, Jose Luis and Pranjal Kumar Rai (2014). *Nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees.* `https://github.com/jlblancoc/nanoflann`.

Goldman, D. and J. Chen (n.d.). In: *Ieee international conference on computer vision (iccv).*

Grupp, Michael (2017). *Evo: python package for the evaluation of odometry and slam.* `https://github.com/MichaelGrupp/evo`.

Hata, Alberto and Denis Wolf (2014). "Road marking detection using lidar reflective intensity data and its application to vehicle localization." In: *17th international ieee conference on intelligent transportation systems (itsc)*. IEEE, pp. 584–589.

Hertzberg, Christoph, René Wagner, Udo Frese, and Lutz Schröder (2013). "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds." In: *Information fusion* 14.1, pp. 57–77.

Höfle, Bernhard and Norbert Pfeifer (2007). "Correction of laser scanning intensity data: data and model-driven approaches." In: *Isprs journal of photogrammetry and remote sensing* 62.6, pp. 415–433.

Jelalian, Albert V (1992). *Laser radar systems.* Artech House on Demand.

Jutzi, Boris and H Gross (2009). "Normalization of lidar intensity data based on range and surface incidence angle." In: *Int. arch. photogramm. remote sens. spat. inf. sci* 38, pp. 213–218.

Jutzi, Boris and Uwe Stilla (2006). "Range determination with waveform recording laser systems using a wiener filter." In: *Isprs journal of photogrammetry and remote sensing* 61.2, pp. 95–107.

Kashani, Alireza G, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson (2015). "A review of lidar radiometric processing: from ad hoc intensity correction to rigorous radiometric calibration." In: *Sensors* 15.11, pp. 28099–28128.

Lehner, Hubert and Christian Briese (2010). "Radiometric calibration of full-waveform airborne laser scanning data based on natural surfaces." In:

Levenberg, K (1944). “Method for the solution of certain problems in least squares siam.” In: *J numer anal* 16, 588–A604.

Levinson, Jesse and Sebastian Thrun (2010). “Robust vehicle localization in urban environments using probabilistic maps.” In: *2010 ieee international conference on robotics and automation*. IEEE, pp. 4372–4378.

Marquardt, Donald W (1963). “An algorithm for least-squares estimation of nonlinear parameters.” In: *Journal of the society for industrial and applied mathematics* 11.2, pp. 431–441.

Muja, Marius and David G Lowe (2009). “Fast approximate nearest neighbors with automatic algorithm configuration.” In: *Visapp (1)* 2.331-340, p. 2.

Phong, Bui Tuong (1998). “Illumination for computer generated pictures.” In: *Seminal graphics: pioneering efforts that shaped the field*, pp. 95–101.

Quigley, “Morgan, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng” (2009). ““ros: an open-source robot operating system”.” In: *“proc. of the ieee intl. conf. on robotics and automation (icra) workshop on open source robotics”*.

Rusu, Radu Bogdan (Oct. 2009). “Semantic 3d object maps for everyday manipulation in human living environments.” PhD thesis. Computer Science department, Technische Universitaet Muenchen, Germany.

Steder, Bastian, Michael Ruhnke, Rainer Kümmerle, and Wolfram Burgard (2015). “Maximum likelihood remission calibration for groups of heterogeneous laser scanners.” In: *2015 ieee international conference on robotics and automation (icra)*. IEEE, pp. 2078–2083.

Truemper, Klaus (1982). “Alpha-balanced graphs and matrices and gf (3)-representability of matroids.” In: *Journal of combinatorial theory, series b* 32.2, pp. 112–139.

Umeyama, Shinji (1991). “Least-squares estimation of transformation parameters between two point patterns.” In: *Ieee transactions on pattern analysis & machine intelligence* 13.04, pp. 376–380.

Wang, Han, Chen Wang, and Lihua Xie (2021). “Intensity-slam: intensity assisted localization and mapping for large scale environment.” In: *Ieee robotics and automation letters* 6.2, pp. 1715–1721.

Xu, Wei, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang (2022). “Fast-lio2: fast direct lidar-inertial odometry.” In: *Ieee transactions on robotics*.

Xu, Wei and Fu Zhang (2021). “Fast-lio: a fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter.” In: *Ieee robotics and automation letters* 6.2, pp. 3317–3324.

Yan, Wai Yeung, Ahmed Shaker, Ayman Habib, and Ana Paula Kersting (2012). “Improving classification accuracy of airborne lidar intensity data by geometric calibration and radiometric correction.” In: *Isprs journal of photogrammetry and remote sensing* 67, pp. 35–44.

Zhang, Lintong, Marco Camurri, and Maurice Fallon (2021). *Multi-camera lidar inertial extension to the newer college dataset*. arXiv: 2112.08854 [cs.RO].

Zhang, Zichao and Davide Scaramuzza (2018). "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." In: *2018 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, pp. 7244–7251.