Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Grunde Nyvoll

Exploiting Lidar Imaging for Robot Perception and Place Recognition

Master's thesis in Cybernetics and Robotics
Supervisor: Kostas Alexis
June 2022

NTNU
Norwegian University of
Science and Technology

Grunde Nyvoll

Exploiting Lidar Imaging for Robot Perception and Place Recognition

Master's thesis in Cybernetics and Robotics
Supervisor: Kostas Alexis
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

In recent years, lidars have seen major technological improvements to deliver ever-increasing point cloud resolution. In addition to the capability of measuring range to objects in the scene, most high-end lidars also output the intensity in the reflected signal. Some lidars can also measure the ambient lighting from the environment. This enables the lidar to act as a 360° imaging sensor, though at a relatively low resolution compared to what can be expected by a normal camera. Recent research has showed promising results in artificially increasing the resolution of an image by deep learning-based methods. Motivated by this, the work in this thesis has explored if it is possible to increase the apparent resolution of real lidar images by applying such methods, which are known as super resolution methods.

Additionally, it has been studied if the intensity and ambient lidar images can be used in combination for visual place recognition by a bag-of-words-based method. To facilitate the collection of real sensor data, a handheld multi-modal sensor rig has been built. The thesis therefore also serves as documentation for the development process of this sensor rig, which will likely be useful for future research within robotic perception.

The results of using ambient and intensity lidar images for place recognition in urban environments are promising, even during reverse revisits and when operating over extended periods of time. The results of applying super resolution methods on real lidar images seems to give reasonable results for range images, though extending it to intensity and ambient images proved to be less effective, which is likely a result of a high degree of noise present in the images.

As a possible way forward to improve the super resolution results, a pipeline has been proposed that can generate realistic looking lidar images with arbitrary resolution based on multiple point clouds, but with lower image noise. The aim for this pipeline is to generate a better lidar image dataset, which can be used to train an improved neural network for the lidar image super resolution problem. The results from testing the image generation pipeline shows promising results for lidar intensity images, which potentially can be used to train a more effective intensity super resolution network in the future.

Sammendrag

I de senere årene har lidarteknologi gjort store fremskritt, som har resultert i stadig høyere punktskyoppløsning. I tillegg til å kunne måle avstanden til objekter, så kan mange høykvalitetslidarer også måle intensiteten i reflekterte signaler. Noen lidarer har også mulighet til å måle omgivelsesbelysningen i et område. Dette gjør at lidaren kan brukes som et 360° kamera, men med en relativt begrenset oppløsning sammenlignet med hva man kan forvente av et vanlig kamera. Nyere forskning har vist lovende resultater i å kunstig øke oppløsningen til et bilde ved hjelp av metoder basert på dyp læring. Motivert av dette, har denne masteroppgaven utforsket om det er mulig å kunstig øke oppløsningen til lidarbilder ved hjelp av slike metoder, som er kjent som superoppløsningsmetoder.

I tillegg til dette, har det blitt undersøkt om lidarintensitetsbilder og lidaromgivelsesbelysningsbilder kan brukes sammen for visuell plassgjenkjennelse ved hjelp av en sekk-med-ord-basert metode. For å samle inn sensordata til å teste metodene, har en håndholdt multimodal sensorplattform blitt bygget. Oppgaven dokumenterer dermed også utviklingsprosessen for denne håndholdte sensorplattformen, som kan bli nyttig for fremtidig forskning innen robotpersepsjon.

Resultatene av å bruke lidaromgivelsesbelysningsbilder og lidarintensitetsbilder for visuell plassgjenkjenning i urbane områder er lovende, selv når man kommer tilbake til samme område i motsatt retning av den man opprinnelig kom fra og når systemet opererer over lengre tid. Resultatene av å bruke superoppløsningsmetoder på ekte lidarbilder gir rimelige resultater for avstandsbilder, men fungerte ikke like godt på lidarintensitetsbilder og lidaromgivelsesbelysningsbilder. Dette er sannsynligvis en konsekvens av at lidarbildene inneholder mye støy.

Som en mulig vei videre for å forbedre superoppløsningsresultatene, har det blitt foreslått en pipeline som kan lage realistiske lidarbilder med vilkårlig oppløsning basert på flere punktskyer, men med mindre bildestøy. Målet med denne pipelinen er å kunne lage et bedre lidarbildedatasett, som kan brukes til å trene et forbedret nevralt nettverk for å oppskalere lidarbilder. Resultatene fra denne lidarbildegenerende pipelinen er lovende for lidarintensitetsbilder og kan potensielt brukes til å trene et forbedret superoppløsningsnettverk for lidarintensitetsbilder i fremtiden.

Preface

This thesis was written as the final project for the 5-year master program in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU) during the spring of 2022.

First and foremost, I would like to express my gratitude to my supervisor, Professor Kostas Alexis for providing valuable guidance during this semester and introducing me to a largely unexplored field within robotic perception. I also want to express special appreciation to PhD candidate Nikhil Vijay Khedekar and the rest of the members at the Autonomous Robots Lab (ARL) at NTNU. Without their help with answering questions I have had, CAD modeling of the sensor rig presented in the thesis and allowing me to borrow the equipment at the ARL, the results from this semester would not have been possible. I also want to thank Leica Geosystems who provided me access to their world spanning GNSS correction service, HxGN SmartNet.

Finally, I wish to thank my friends and family for their continued support and encouragement up until this point.

Grunde Nyvoll

Trondheim, June 2022

Table of Contents

List of Figures

List of Tables

Acronyms

ANN Artificial Neural Network. 20, 24

ARL Autonomous Robots Lab. v, 46

BoVW Bag of Visual Words. 9, 10, 12, 28, 64

BRIEF Binary Robust Independent Elementary Features. 9

CAD Computer-Aided Design. v, 46

CNN Convolutional Neural Network. 21

FAST Features from Accelerated Segment Test. 8, 9

GigE Gigabit Ethernet. 48

GLONASS GLObalnaja NAvigatsionnaja Sputnikovaja Sistema. 14, 48

GNSS Global Navigation Satellite System. v, xiv, xv, 14–16, 46, 48–54, 61, 63

GPS Global Positioning System. 5, 14, 15, 48

HR High-Resolution. 16–19

HTTP HyperText Transfer Protocol. 52

ICP Iterative Closest Point. xiv, 13, 42–44, 59

IMU Inertial Measurement Unit. 46, 48, 59

ISR Image Super Resolution. 2

lidar light detection and ranging. 1

LR Low-Resolution. 16–18

MAD Median Average Deviation. 57

MAE Mean Average Error. 57

NMEA National Marine Electronics Association. xiv, 52, 53

NRTK Network Real-Time Kinematic. 15, 16, 51

NTNU Norwegian University of Science and Technology. v, 46

NTRIP Networked Transport of RTCM via Internet Protocol. xiv, 51–53

NUC Next Unit of Computing. 46–48

ORB Oriented FAST and Rotated BRIEF. 9, 28, 30, 54, 55

PSNR Peak-Signal-to-Noise Ratio. 19, 56, 58, 85, 86

ReLu Rectified Linear Unit. xiii, 24, 33

ROS Robot Operating System. xiv, 51–53

RTCM Radio Technical Commission for Maritime Services. xiv, 51–53

RTK Real-Time Kinematic. xiv, xv, 15, 46, 48, 51–53, 61, 63

SIFT Scale-Invariant Feature Transform. 9

SISR Single Image Super Resolution. xii, 17–20, 31, 40, 88

SLAM Simultaneous Localization And Mapping. 7, 8, 13, 14, 28, 104

SSIM Structural Similarity Index Measure. 19, 20, 58, 85, 86

TF-IDF Term Frequency–Inverse Document Frequency. 10, 54

USB Universal Serial Bus. xiv, 48, 49, 52, 53

VPR Visual Place Recognition. xv, xvi, xxi, 8, 12, 28, 30, 54, 55, 61, 64–69

Chapter 1

Introduction

1.1 Motivation

In the domain of autonomous robotics, light detection and ranging (lidar) sensors are often a core part a robot's perception system. A lidar is an optical sensor that leverages lasers to measure the distance to objects in the scene. By using multiple rotating lasers at varying heights, a high-end lidar can output a fairly detailed point cloud as shown in Figure 1.1. Also, many lidars do not only output

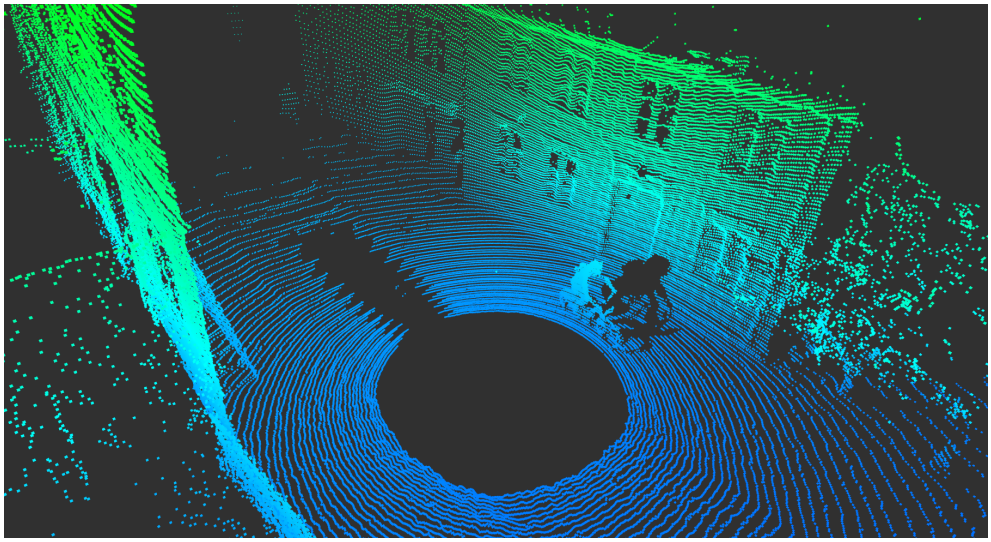


Figure 1.1: Example of a point cloud from a lidar.

the detected range, but also the intensity in the reflected signal, which is highly correlated with the reflective properties of the object it reflected from [1]. Quite recently, Ouster [2] introduced a lidar that is not only able to measure range and intensity, but also the ambient lighting from external sources such as the sun, effectively making the lidar act as 360° camera [3] as shown in Figure 1.2. To a human observer it would be easier to identify the location based on a combina-

(a) Intensity image

(b) Ambient image

(c) Range image

Figure 1.2: The intensity, ambient and range image outputted by an Ouster lidar.

tion of the images, but most robotic perception systems today, only use the range information from the lidar. This poses the question if this additional information can be leveraged by a robot for improved place recognition.

However, it is apparent that the vertical resolution of the images is not comparable to what you would expect from a normal camera. The reason is that the vertical resolution of the images is limited by the number of lasers, called channels, for the given lidar. This makes it harder to perceive the environment, both in terms of recognizing the location, but also to distinguish the smaller structures. Buying a lidar with a higher channel-number can also be a rather expensive endeavour, usually costing several thousand dollars.

In recent years there has been active research within deep learning to convert low resolution images into higher resolution images, a problem known as Image Super Resolution (ISR). Some neural networks designed to tackle this problem have shown very promising results in areas such as medical and surveillance imaging [4, 5]. A key question is if these advancements can be extended further to the lidar domain.

1.2 Focus Topics

Motivated by the new measurement capabilities of lidars and the developments in deep learning for the image super resolution problem, the work to be presented here has focused on the following topics:

- Use both the intensity and ambient images from a lidar to attempt to improve robotic place recognition.
- Applying neural networks to explore the possibility of increasing the apparent resolution of the range, intensity and ambient image from a real lidar.

1.3 Thesis Structure

The thesis is structured in the following way. Chapter 2 presents the theory which is used as a baseline for the rest of the thesis and it is divided into two parts. The first half covers the place recognition problem, in particular the visual place recognition problem. The second half of the chapter covers the super resolution problem, where we mainly focus on the image super resolution problem. Chapter 3 is the methodology, which begins by covering a new lidar place recognition algorithm. Then, the chapter presents a pre-existing lidar super resolution pipeline, which the work in this thesis has built upon. Afterwards, the chapter goes into detail about how the lidar super resolution pipeline has been adapted to range, intensity and ambient lidar images. Subsequently, the chapter covers a new method for generating lidar images, that can be used to train a super resolution network. Then the chapter covers the development process of a sensor rig, which has been built, tested and used to collect data for the thesis. The end of the chapter covers the experimental testing procedure of the place recognition algorithm, lidar super resolution pipeline and the lidar image generation pipeline. Chapter 4 covers the results of the experiments from Chapter 3, with a discussion of the results. Finally, Chapter 5 lists some concluding remarks and recommendations for future work.

Chapter 2

Background

This chapter will go into detail about two different topics related to lidar-based robotic perception: place recognition and super resolution. The chapter begins with a common background related to lidars and their adaption in robotic systems. Starting from Section 2.3, the chapter is divided into two parts. The first half of the chapter covers the place recognition problem and some theory related to performance evaluation of place recognition algorithms using GPS. The second part of the chapter, starting at Section 2.7, covers theory related to deep learning-based super resolution. The theory presented here should be detailed enough so that a reader without specific knowledge about the place recognition problem and super resolution, can be able to follow it.

2.1 Lidars in Robotics

Within the last decade there has been large advancements within 3D lidar technology, offering improved resolution, precision and significant cost reductions [6]. This has resulted in 3D lidars being adapted for a multitude of robotic systems, including legged robots [7, 8], flying drones [9] and unmanned ground vehicles [10]. Yet, long range 3D lidar is still a relatively expensive technology, which has restricted their use in consumer products. On the other hand, 2D lidar has in the recent years also seen considerable price reductions and can at the time of writing be bought for less than $100 dollars [11]. This has made the lidar sensor starting to find its way into robotic household products, like robot vacuum cleaners [12], but the adaption in the consumer market is still in its infancy. In this thesis we will though restrict our focus to long range 3D lidar.

2.2 Ouster Lidar

As mentioned in Section 1.1, most lidars on the market today are capable of outputting both range and intensity measurements. However, the definition of intensity varies both within the scientific community and in the industry but can

broadly be defined as the strength in the returned signal [1]. In order to more precisely understand the properties of the range, intensity and ambient images, here defined as the raster bands of the lidar, we will narrow down our focus to one specific series of commercial lidars, which is the OS0 by Ouster[1]. This is also because the ambient image is only available for this series of lidars. Note that since this is a proprietary technology, we will have to base the description based on what Ouster have published on their website[2].

The Ouster lidar is a semiconductor-based lidar, where all the laser beam detectors are placed on single chip. These detectors are able to count individual photons at the 850 nm operating wavelength of the lidar. Since there are other significant noise factors at this wavelength, in particular the sun, the detectors will measure a combination of photons originating from the lidar itself and external sources as illustrated in Figure 2.1. The photons that the lidar estimates to originate from itself, is what the Ouster lidar leverages to generate the intensity image. Meanwhile, the photons that the lidar believes originate from other sources, are used to create the ambient image. To generate the range image, the lidar works on the principle of time-of-flight. By measuring the time it takes from a lidar pulse is emitted to when it is received, the distance can be calculated, since electromagnetic waves travel at the speed of light [13].

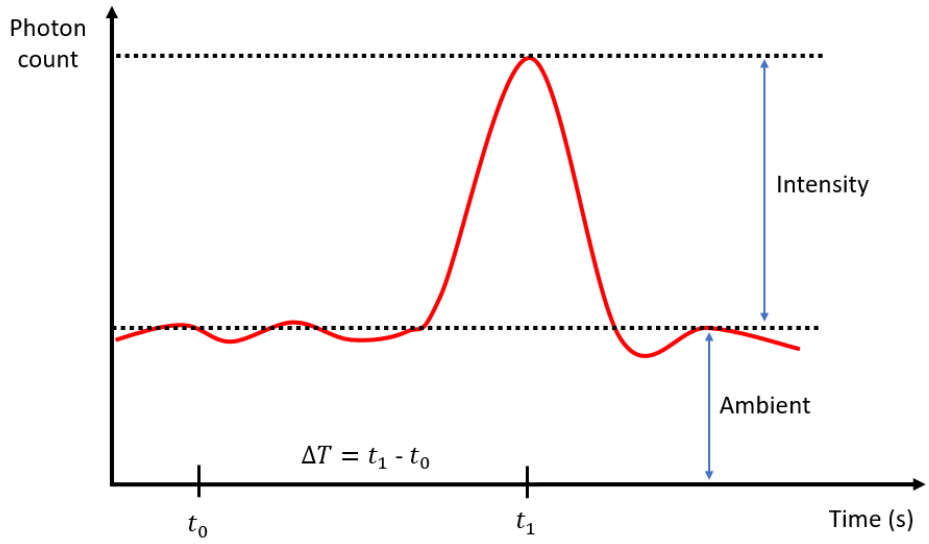
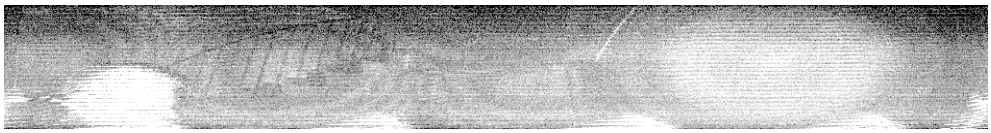


Figure 2.1: The relation between intensity and ambient data for the Ouster lidar. The intensity data corresponds to the photons originating from the lidar itself, while the ambient data corresponds to photons from external sources. By emitting a laser pulse at time t_0 and measuring the time it takes until it returns at time t_1, the distance can be calculated. Figure adapted from [14].

[1] https://ouster.com/products/scanning-lidar/os0-sensor/
[2] https://ouster.com/blog/how-multi-beam-flash-lidar-works/

A challenge with the ambient images, is that since they are dependent on external lighting, they become very susceptible to poor illumination conditions as can be seen in Figure 2.2. Meanwhile, since the intensity image is based on laser beams from the lidar itself, this image would be virtually unaffected by such conditions.

(a) Ambient image in the shadow of a building.

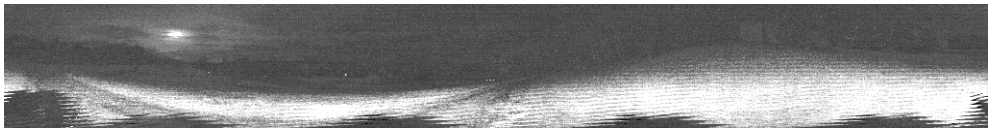


(b) Ambient image indoor.

Figure 2.2: Examples where the ambient image from an Ouster lidar becomes noisy due to inadequate external lighting.

On the other hand, when there are good lighting conditions, the ambient images can be relatively feature rich. This can for instance be seen in Figure 2.3, where buildings in the far background are clearly more visible in the ambient image than in the intensity image.

(a) Ambient image



(b) Intensity image

Figure 2.3: An example where the ambient image (a) provides more details than the intensity image (b) in good lighting conditions.

2.3 Place Recognition

In the domain of robotic perception, place recognition is the problem of identifying when a previously seen location is revisited [15]. Place recognition is an essential part of the Simultaneous Localization And Mapping (SLAM) problem, which is one of the most researched topics within the robotic community. The task to be solved in SLAM, is for a robot to build a map of an unknown environment and

localize itself in this map at the same time [16]. A core challenge with SLAM is to mitigate drift in the estimate. Here is where place recognition, also known as loop-closure detection, plays a vital role. By detecting a loop-closure, the accumulated drift can be identified and compensated for [17]. However, place recognition can be challenging, especially due to perceptual aliasing, where different locations appear visually similar. To make matters worse, even a single incorrectly identified loop-closure can be detrimental if it is used in a SLAM algorithm [15].

Many methods have been proposed to tackle the place recognition problem and most of them can be put into one of two categories; visual-based methods and point cloud-based methods [15, 18]

2.4 Visual-Based Place Recognition

Visual Place Recognition (VPR) is the task of identifying a previously visited location based on images. It has received great attention within the academic community due to the general availability of cameras and the feature richness of images [15]. VPR can be extremely difficult, as it has to account for multiple different factors, including changes in lighting, different viewpoints and changing weather conditions [19]. Yet, since many state-of-the-art SLAM methods use cameras as the primary sensor [20–22], visual-based place recognition is an essential task that needs to be solved.

There is also a large variability in the assumptions made by the different VPR methods. In a pure *appearance-based* VPR method, the algorithm will typically search for potential loop-closures among all previously visited locations [15]. The advantage of the pure appearance-based methods is that they can detect loop-closures even if the estimated robot position has drifted significantly, which can be the case e.g. when operating in large-scale environments [23].

A potential problem with the appearance-based VPR methods, is that the search space can in some cases become so large that it becomes challenging to meet real-time demands [24]. In this case, a *geometric* or *hybrid* VPR method might be more suitable, where the VPR algorithm also make use of an estimated position of the robot and only searches for potential loop-closures locally. Another advantage is that leveraging an estimate of the position of the robot can often improve the place recognition accuracy, given that the estimated position has not drifted too much [25].

2.4.1 Feature Detectors and Descriptors

A principal component of most VPR algorithms is that images can often by broken into a set of points of interest, or *keypoints*. This is an essential idea across the computer vision domain, and an algorithm that can find these keypoints is called a *feature detector* [26–28]. Many of the feature detection algorithms, such as the Features from Accelerated Segment Test (FAST) detector [29], actively seek for corners in an image, since these are usually easy to track.

Feature detectors are seldom used without combining them with a *feature descriptor*, which can describe the keypoints [30–32]. Feature descriptors can be placed into one of two categories, *floating-point* descriptors and *binary* descriptors. Floating-point descriptors, such as the widely popular Scale-Invariant Feature Transform (SIFT) descriptor [33], often shows superior results in terms of accuracy, but at a cost of usually being more computationally expensive [26].

Binary descriptors such as the Binary Robust Independent Elementary Features (BRIEF), usually operate directly on individual pixel intensities in patches of an image and encodes them into binary sequences [34]. However, even though BRIEF is efficient to compute, it is sensitive to both changes in scale and image rotations.

A popular binary descriptor based on BRIEF, which attempts to alleviate these issues, is the Oriented FAST and Rotated BRIEF (ORB) descriptor [35]. The ORB descriptor leverages the FAST keypoint detector to rotate the BRIEF descriptors, so that they become invariant to in-plane rotations. To address the problem with scale, ORB uses an image pyramid with multiple levels, where the first level corresponds to the original image and the succeeding levels are created by downsampling the image from the previous layer by a given factor. This effectively simulates viewing the same scene at multiple different scales. By detecting feature points at each of these levels, the ORB detector becomes partly invariant to changes in scale [35]. A major advantage of the binary descriptors is that they can be efficiently compared with each other by simply counting the number of differing bits between their binary sequences, a metric known as the Hamming distance [26].

2.4.2 Bag of Visual Words

Many visual place recognition methods leverage a technique known as bag-of-words [23, 36–38]. In the domain of computer vision, bag-of-words is also referred to as Bag of Visual Words (BoVW). The central idea is that by extracting feature descriptors from an image, the descriptors can be converted into a more abstract representation known as visual words. How a visual word is found from a feature descriptor, vary between different BoVW methods, though one of the most common approaches is to compute feature descriptors on a large set of representative images that have been collected in advance. The feature descriptors can then be grouped into clusters using e.g. the K-means algorithm [39], which is illustrated in Figure 2.4. The K-means algorithm seeks to iteratively separate a set of unlabelled points into K clusters and retrieve a set of K means corresponding to the center of each cluster.

After training the K-means model on the set of feature descriptors, these will constitute a *visual vocabulary*. The visual vocabulary is what defines what the visual word for a given feature descriptor is, by assigning it to the nearest mean value in the K-means model. By counting the number of feature descriptors that are assigned to each visual word, a histogram that describes an image can be generated. The number of bins in the histogram will correspond to the number

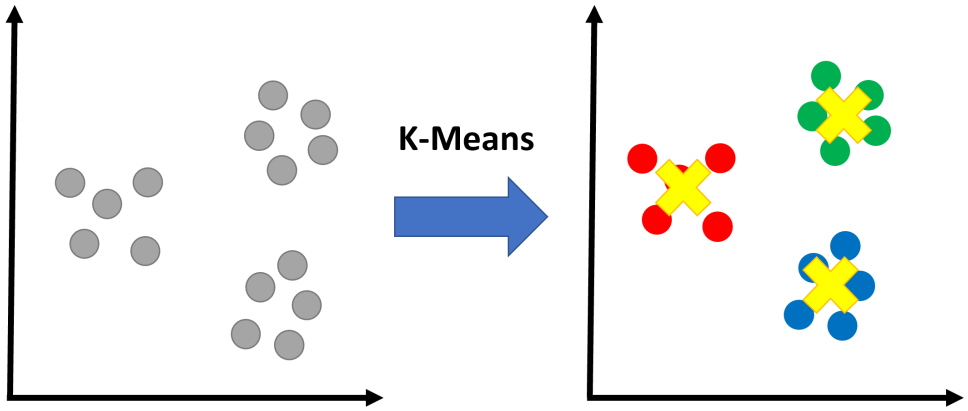


Figure 2.4: Example of the K-means algorithm, where the goal is to separate a set of unlabelled points (grey) into K clusters. The outcome of a successful execution of the algorithm is a set of "means", here represented by cross-marks (yellow), corresponding to the center of each cluster.

of means in the K-means model. The histograms, which are usually represented as vectors, can then be compared with the histogram of other images to search for similarity [36]. A simplistic example of the visual word process is given in Figure 2.5.

However, some visual words will inevitably appear more often and are therefore less distinctive for a given scene. To compensate for this, a function that weights each bin in the histogram is used, where the most used weighting function is the Term Frequency–Inverse Document Frequency (TF-IDF). In TF-IDF a bin x_i is assigned a weight t_i, which is computed as

$$t_i = \frac{n_{id}}{n_d} \log(\frac{N}{n_i}), \tag{2.1}$$

where n_d and n_{id} are the total number of visual words in image d and the number of occurrences of visual word i in the image respectively, N is the total number of images and n_i is the number of occurrences of the visual word i in all the images [36]. This effectively increases the weight of visual words that appear frequently in a given image, but rarely occur in the rest of the images.

Binary Bag of Visual Words

Even though BoVW has become a popular choice for loop-closure detection, it can be both computationally costly and require a significant amount of storage. In order to alleviate these problems, a binary descriptor-based BoVW method, *DBoW*, was proposed by Gálvez-López et al. [40]. In DBoW, the visual vocabulary is built up as a tree structure as shown in Figure 2.6. In the same way as before, we begin by extracting binary feature descriptors from a representative set of images.

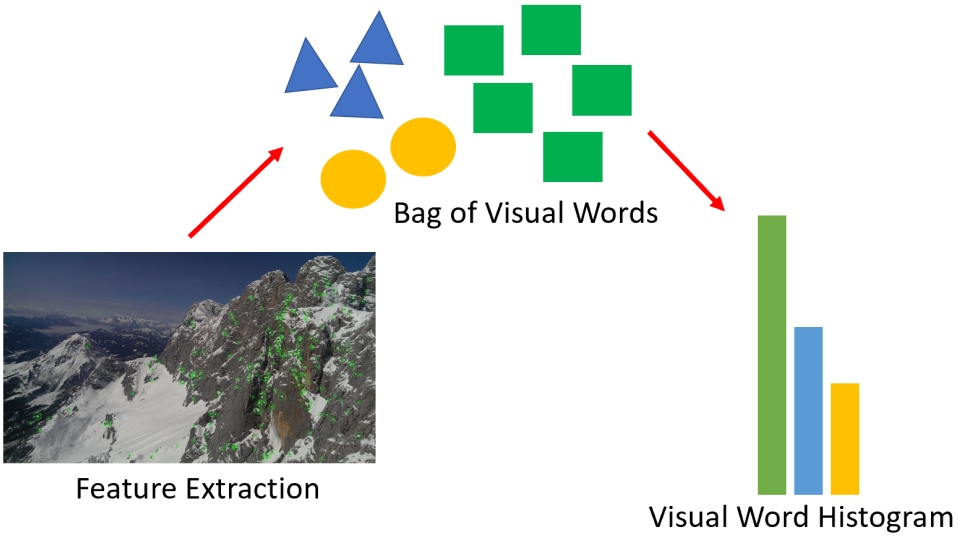


Figure 2.5: Example of the bag of visual words approach. Features from an input image are first extracted using a feature detector and a feature descriptor. The extracted features are then converted into visual words by leveraging a precomputed visual vocabulary. The frequency of each visual word is counted to convert the bag of visual words into a visual word histogram, which can be used to search for similar images.

Starting at the top of the tree at the root node, we cluster the feature descriptors into K bins by using the K-medians algorithm [41], which is a variation of K-means. The number of bins K corresponds to the *branching factor* of the tree.

After the initial clustering, feature descriptors that ended up in the same bin are further clustered into K new bins, which creates a new layer in the tree. The number of times this process is repeated is the *depth level* of the visual vocabulary tree. The nodes at the final layer of the tree, which are called leaf nodes, will correspond to the visual words of the vocabulary [40].

To convert a binary feature vector v_1 into a visual word using the binary vocabulary, we start at the root of the vocabulary tree and select the bin which minimizes the Hamming distance. This procedure is repeated at each layer of the vocabulary tree until we arrive at a leaf node, in which case the visual word is determined.

To estimate the similarity between two binary feature vectors v_1 and v_2, Gálvez-López et al. propose to use a scoring function defined as

$$s(v_1, v_2) = 1 - \frac{1}{2}\left|\frac{v_1}{|v_1|} - \frac{v_2}{|v_2|}\right|, \tag{2.2}$$

where $s(v_1, v_2)$ is called the L_1 score and $|\cdot|$ denotes the L1 norm [40].

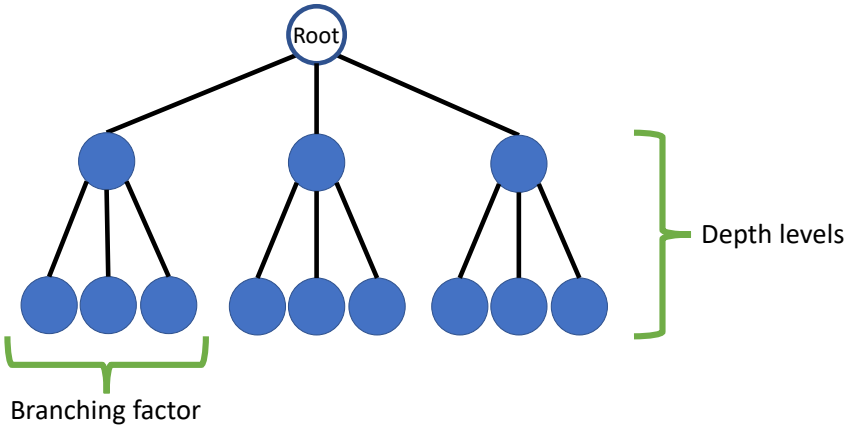


Figure 2.6: Example of a tree data structure. The number of nodes that are connected to the same node one level up in the hierarchy is called the tree's branching factor. The depth level of the tree is the same as the longest path from a node in the bottom of the tree to the root node at the top of the tree. Image adapted from [40]

2.4.3 Lidar Images

Since cameras have been the dominant sensor for high-resolution imagery, VPR has often implicitly been referring to camera images. Nevertheless, with the recent advancement with increased lidar resolution, some researchers have started to investigate using traditional VPR methods for lidar intensity images [42]. In [43] Shan et. al proposed a new VPR method specifically for lidar intensity images. By using conventional image feature descriptors for lidar intensity images, combined with a BoVW approach, their method can retrieve potential loop closure candidates. To reduce the number of false positive loop-closures, Shan et. al leverage the raw point clouds to validate the potential loop-closure candidates.

2.5 Point Cloud-Based Place Recognition

Compared to VPR, place recognition based on point clouds is still a relatively unexplored field. One of the reasons for this, is that point cloud-based place recognition is extremely challenging. However, point cloud-based place recognition has in the the recent years garnered increasing amount of academic attention [18, 44–47]. This comes from the fact that one of the most common 3D point cloud sensors, the lidar, has some very desirable characteristics for the place recognition problem. One of these factors is that, if we make an exception for the ambient data, the lidar point clouds are virtually unaffected by changes in illumination. Lidars

usually also have a much larger field of view than their camera counterpart, which reduces the likelihood of having non-overlapping frames between revisits to the same location [48].

Some point clouds-based place recognition methods attempts to perform global place recognition by encoding specific points, such as the highest visible points [48], or points with a large intensity reading [47], into a unique place signature. Meanwhile, some of the newer methods for global point cloud-based place recognition leverages deep learning-based methods to extract local features from the point clouds, which are then clustered into global descriptors [18, 49]. However, deep learning-based methods usually requires a lot of training data and can be too computationally expensive to be used in real-time setting.

Many point cloud-based place recognition algorithms that are actively used today, only search for loop closures in a local area. These methods are usually dependent on an estimated position and point cloud map from a SLAM pipeline, where they try to align the current point cloud with the map at the current position estimate [50, 51]. However, how this point cloud alignment is performed, varies between the different methods.

2.5.1 Point Cloud Alignment

Local Methods

A widely used algorithm for aligning two point clouds is the Iterative Closest Point (ICP) algorithm [52]. One of the requirements for ICP, is that it is given an initial estimate of the transformation that aligns the two point clouds, which categorizes ICP as a *local* point cloud registration method. ICP works by leveraging the initial transformation to match the closest points in each of the two point clouds and calculating a new transformation that minimizes a cost function, typically the sum of squared differences, with respect to these matches. The resulting transformation is used to find a new set of point cloud matches that have to be optimized. By repeating this procedure of matching and optimization, ICP can often result in a highly accurate point cloud alignment, if the initial transformation is sufficiently close to the solution and if the underlying problem is sufficiently constrained.

Global Methods

A commonly faced challenge when using local registration methods such as ICP, is that they are usually highly dependent on a good initial transformation to avoid becoming stuck in a local minimum. Therefore, there have been proposed multiple methods that seek to perform point cloud alignment without any prior knowledge about the initial transformation, which are called *global* point cloud registration methods. Many of the global point cloud registration methods works by extracting and matching features between the point clouds [53, 54], but these methods are usually highly reliant on the accuracy in the feature matching, which can be

a problem when operating under high noise conditions. Also, these methods usually only utilize the range information in the point cloud, while other sources of information, like intensity and ambient, remains unused.

Quite recently, a new global point cloud registration method called PHASER was proposed in [55] by Bernreiter et al., which seeks to alleviate the problems with robustness by utilizing the additional lidar information. To enable this, the authors leverage the Fourier transform to fuse the information from different modalities, such as range, intensity and ambient.

Still, global point cloud registration methods usually do not result in as tight alignment as a well-initialized local method. For this reason, the resulting transformation from a global registration method is in practise often used as an initial transformation for a local method to further refine the alignment [56].

2.6 Place Recognition Verification

A challenge with evaluating place recognition and SLAM algorithms, is how the ground truth can be found. In public datasets that are captured in outdoor environments, it is common to rely on satellite-based positioning using the Global Navigation Satellite System (GNSS) for this purpose [57, 58]. GNSS is a technology that was first developed for military applications in the 1960-70s. Today there are several independent satellite systems in operation, such as Galileo, GLONASS, BeiDou and the most commonly known, the Global Positioning System (GPS) [59].

2.6.1 Differential GNSS

A common problem with pure satellite-based GNSS positioning is that the there are several noise factors that can deteriorate the accuracy of the position estimate. These include for instance small errors in the internal clocks of the satellites and their estimated position. Another significant factor is that the electromagnetic waves from the satellites can be slowed down when travelling through the atmosphere. This limits the accuracy of the standard GNSS positioning to a few meters, which can be insufficient for some applications [60].

One way to improve the accuracy of the GNSS positioning, is to use differential GNSS. For differential GNSS to be possible, a separate GNSS receiver with a known position - a reference station, is needed [61]. The basic premise of differential GNSS is that some of the noise factors are slowly varying and remain relatively similar within a given area. Since the position of the reference station and the satellite is known, the reference station can compare the true range to the satellite with the perceived range based on the GNSS signals to estimate the range error. Since other GNSS receivers in the area are likely subject to similar noise conditions as the reference station, it is reasonable to assume that they will experience a similar ranging error to the same satellite. Thus, by broadcasting the

estimated range error for each satellite from the reference station to the GNSS receivers in the area, it is possible to reduce the positioning error [61].

2.6.2 Real-Time Kinematic

Real-Time Kinematic (RTK) positioning is one form of differential GNSS, but there are a few differences from the classical differential GNSS approach. In the domain of RTK positioning, the reference station is referred to as a *base station*, while the GNSS receiver that receives the corrections is called a *rover* [62]. Unlike classical GNSS, the distance to each satellite is not calculated based on the data encoded in the GNSS signal itself. Instead RTK GNSS leverages that the carrier wave for a given GNSS satellite operates at a known frequency, i.e. a GPS satellite in the L1 band has a carrier wave frequency of 1575.42 MHz [63]. We know that the relation between wavelength and frequency of a periodic wave is given by

$$\lambda = v/f, \tag{2.3}$$

where λ is the wavelength, v is the phase speed of the wave, while f is the wave's frequency. Since GNSS signals are electromagnetic waves, they propagate at the speed of light, $c = 300,000$ km/s, so the wavelength of the carrier wave can be calculated as

$$\lambda = c/f = \frac{300,000\,\text{km/s}}{1575.42\,\text{MHz}} \approx 19\,\text{cm}. \tag{2.4}$$

As the electromagnetic wave travels from the satellite to the GNSS receiver, the wave will have completed N whole wave cycles and one partially completed cycle corresponding to the phase of the carrier wave, ϕ. If we can determine the number of completed cycles N and measure the carrier wave phase ϕ, then the distance to the satellite can be calculated as $N \cdot \lambda + \phi \cdot \lambda$, since the wavelength λ is known. The problem is that determining the number of completed cycles N is not trivial, and it is known as the *integer ambiguity resolution* problem [64].

However, if a base station measures the phase of the carrier wave, then the number of completed cycles can be determined. This comes from the fact that the location of the base station and the satellite is known and therefore also the range to the satellite. By sending the measured carrier phase shift from the base station to the rover, together with the position of the base station itself, it may be possible for the rover to calculate the number of completed cycles N. One assumption for this to be possible, is that the rover and base station have at least five satellites in common. If the solution is found, it is referred to as a fixed solution, and it usually has an accuracy better than 1-2 cm [65].

2.6.3 Network Real-Time Kinematic

An obvious disadvantage of RTK compared to classical GNSS, is the requirement of having access to a base station in close vicinity and being able to communicate reliably with it. A popular solution to the former problem is Network Real-Time

Kinematic (NRTK), where a user can gain access to a large network of base stations that the rover can communicate with over the internet [66]. In a NRTK system, the internet connection is used as a two-way communication link between the base and the rover. This enables the rover to inform the GNSS correction service about its current position, so that the closest base station(s) can be assigned to the rover, while also making it possible to improve the correctional data by mathematical modeling of distance dependent noise errors [67].

2.7 Super Resolution

The term *super resolution* was first coined by Gerchberg in 1974 to describe the goal of recovering signals with a higher resolution than the theoretical limit in an optical system, which is known as the diffraction limit [68]. In later literature, super resolution is divided into two different categories; optical super resolution and geometrical super resolution, where optical super resolution is what Gerchberg originally referred to [69]. Meanwhile, geometrical super resolution is to increase the resolution of an image beyond the resolution imposed by a given digital detector array, which is the super resolution definition we will be using here [70].

The super resolution problem has a long history within the domain of medical imaging, where high-resolution images is often necessary to generate accurate 3D models for adequate diagnosis and treatment. However, acquiring high-resolution medical images is usually very time consuming and expensive or not possible at all, which has resulted in the adaptation of super resolution techniques to artificially increase the apparent resolution of the images [71]. Super resolution has also become an essential tool in surveillance imaging analysis, where the typically low image resolution can make object detection extremely challenging [72].

Although super resolution is most commonly associated with the problem of super-resolving images, the term super resolution covers a much wider number of problems. One of these problems, which is particularly relevant for lidars, is the point cloud super resolution problem [73, 74]. We will therefore start here by going through the point cloud super resolution problem and how it can be connected to the problem of super-resolving images.

2.7.1 Point Cloud Super Resolution

The point cloud super resolution problem is the task of acquiring a High-Resolution (HR) point cloud from a Low-Resolution (LR) input cloud as illustrated in Figure 2.7. This problem has received a lot of attention by the academic community, due to point clouds often being sparse in nature [74, 75]. Early attempts on solving this problem were usually optimization-based and often made explicit assumptions on the structure of the underlying point cloud [76, 77]. However, with the advancements in deep learning, there has been a general trend towards data-driven

methods, leveraging deep neural networks for the point cloud super resolution problem [73, 74, 78]

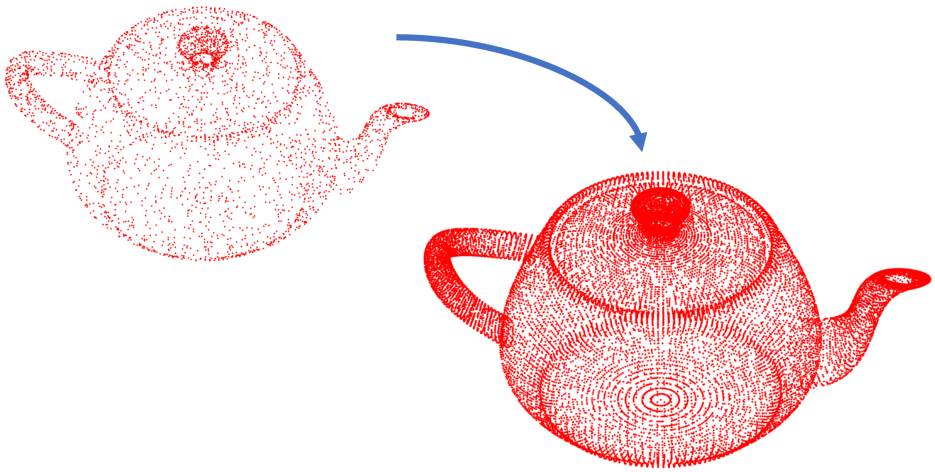
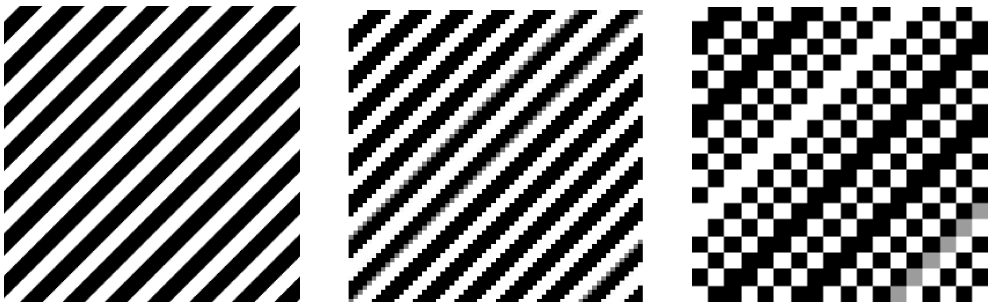


Figure 2.7: Illustration of the point cloud super resolution problem, where the goal is to recover a high-resolution point cloud from a low-resolution input point cloud.

Although some of the deep-learning based methods have demonstrated remarkable results for super-resolving point clouds, they will usually not preserve sensor specific characteristics, e.g. the ring-pattern that is present in most lidar point clouds, like the one we saw in the introduction in Figure 1.1. One of the few methods that specifically attempts to address the problem of super-resolving lidar point clouds, while still preserving this ring-pattern is presented by Shan et al. in [79]. Here the authors propose to project the point cloud onto a 2D image to generate a range image. Then instead of super-resolving the point cloud directly, the authors propose to super-resolve the range image and project it back into 3D space to retrieve a densified version of the original lidar point cloud. In order to super-resolve the range image, the authors leverages techniques from one of the most well known super resolution problems, which is the *single image super resolution* problem.

2.7.2 Single Image Super Resolution

Single Image Super Resolution (SISR) is the problem of recovering a HR image from a single LR image [80]. The main challenge with SISR is that there are multiple HR images that could originate from the same LR image, so the solution is not unique [81]. The reason for this can be explained by the well-known Nyquist-Shannon sampling theorem. This states that it is possible to recreate a signal given that the sampling frequency f_s is at least twice as high as the highest frequency



(a) Original **(b)** Oversampled **(c)** Undersampled

Figure 2.8: Illustration of the aliasing effect in 2D. The original image of a 2D sine wave (a) is sampled at above the Nyquist frequency (b) and below the Nyquist frequency (c). When sampling below the Nyquist frequency, the image becomes clearly distorted with patterns that were not present in the original image.

component f_c in the signal

$$f_s \geq 2f_c = f_N, \tag{2.5}$$

where the minimum sampling frequency f_N is known as the Nyquist frequency [82]. If this criterion is not met, then the recreated signal will be subject to aliasing, which is illustrated in Figure 2.8. Since the LR images for a given super resolution problem do in general not fulfill the Nyquist criterion, they will be subject to aliasing, which causes information to be lost [83].

Degradation Model

The reverse operation of SISR, that is going from a HR image to a LR image, is also an important topic. A model that defines a mapping from the HR image to the LR image is called a degradation model. An illustration of how the degradation model is linked to the SISR problem is shown in Figure 2.9. If the SISR method assumes that the degradation model is known a priori, then it is categorized as a non-blind method, while in the opposite case it is categorized as a blind method [84]. For a non-blind SISR method to generalize to real data, it is crucial that the selected degradation model matches the degradation in the real data well.

A well-known degradation model is to apply bicubic downsampling on the high-resolution image [85], but this degradation model has proved to usually be too unrealistic, which often results in poor generalization to real images. A more commonly used degradation model, which aims to better represent the degradation that can be expected in real images, is defined by

$$LR = (HR \circledast \mathcal{K}) \downarrow_S + \mathcal{N}, \tag{2.6}$$

where a HR image is first convolved with a blur kernel $\mathcal{K}$ and downscaled by a factor s by the downsampling operator $\downarrow_S$. Finally Gaussian noise $\mathcal{N}$ is added to produce the LR image [84]. Although this model can in some cases result in better

generalization to real images than a bicubic degradation, it still requires that the selected blur and noise levels properly represents what can be expected in the real data [84].

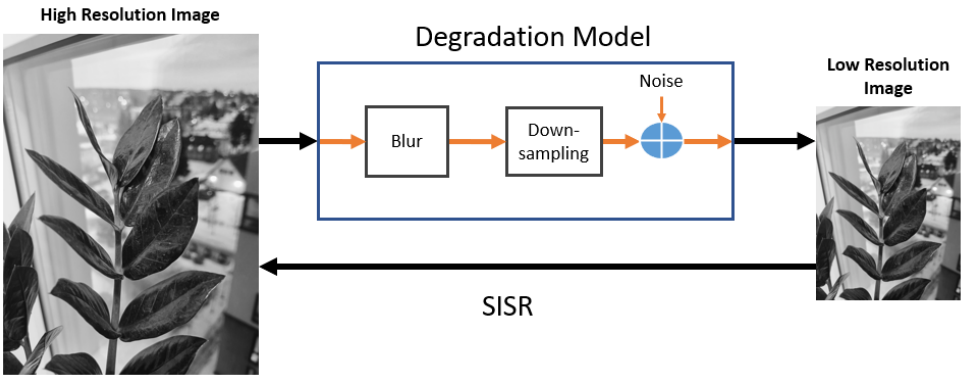


Figure 2.9: Illustration of the Single Image Super Resolution (SISR) problem. The low-resolution image is assumed to originate from a higher resolution image that has been transformed to a low-resolution image by a degradation model. Recovering the high-resolution image from the low-resolution image is known as the SISR problem. Figure adapted from [81].

2.7.3 Image Super Resolution Metrics

A challenge with image super resolution, is how to quantitatively evaluate the reconstruction quality. One of the most commonly used metrics is the Peak-Signal-to-Noise Ratio (PSNR) metric, where for an image pair (x, y) with dimensions $N \times M$, the PSNR value is given as

$$PSNR = 10 \log_{10} \left(\frac{(MAX^n - 1)^2}{MSE(x, y)} \right), \tag{2.7}$$

where MAX^n is the maximum value for an n bit image and

$$MSE(x, y) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{i=j}^{M} (x_{i,j} - y_{i,j})^2, \tag{2.8}$$

where MSE is the mean squared error [86]. One challenge with this metric, is that it assumes that that the original HR image is noise free, and it is commonly criticized for being unable to properly represent the structure and visual quality of the image. To address the two latter issues, Wang et al. [87] proposed the Structural Similarity Index Measure (SSIM) which models the perceived distortions by three different comparison factors calculated over various patches of the image pairs. These three comparison factors are called the luminance $l(x, y)$, contrast $c(x, y)$ and structure $s(x, y)$.

Let μ represent the average pixel intensity (luminance) of a given image patch, then the luminance comparison is defined as

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{2.9}$$

where C_1 is a small non-zero constant which is added for numerical stability. Let σ_x and σ_y be the covariances of the corresponding patches of image x and y respectively, then the contrast comparison is defined as

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{2.10}$$

where C_2 is a small non-zero constant. Before calculating the structure comparison, the corresponding patches of image x and y first have to be standardized by

$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \tag{2.11a}$$

$$\hat{y} = \frac{y - \mu_y}{\sigma_y}, \tag{2.11b}$$

where $\hat{x}$ and $\hat{y}$ are the standardized image patches. The structure comparison of the corresponding standardized images is then defined as

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \tag{2.12}$$

where again C_3 is small non-zero constant and

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y). \tag{2.13}$$

The SSIM is then defined as weighted product of these terms as

$$SSIM(x,y) = [l(x,y)]^\alpha \cdot [c(x,y)]^\beta \cdot [s(x,y)]^\gamma, \tag{2.14}$$

where α, β and γ are weighting exponents which can be tuned [87]. Usually, we have that $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$, where the SSIM can be simplified to

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \tag{2.15}$$

2.8 Single Image Super Resolution Networks

Multiple different Artificial Neural Network (ANN) architectures have been proposed to tackle the SISR problem. Two of the most common approaches are the

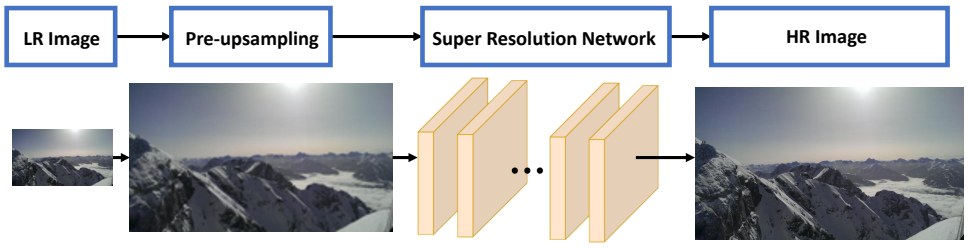


Figure 2.10: Example of the pipeline for a pre-upsampling based super resolution network. Initially, the low-resolution (LR) image is upscaled, by for instance bicubic interpolation, before it is passed through the rest of the super resolution network to produce the high-resolution (HR) image.

pre-upsampling and *post-upsampling* based networks [85, 88, 89]. In a pre-upsampling based network, the low-resolution image is initially upscaled to the same size as the high-resolution image using e.g. bicubic interpolation, before it is fed into the rest of the network [85] as shown in Figure 2.10.

However, a problem with pre-upsampling the image is that the computational cost in a neural network usually increases drastically with the input dimensions. This can therefore result in significant constraints on the depth of the network architecture [90, 91].

To address this issue, methods leveraging post-upsampling were proposed, where the low-resolution image is used directly as the input [92]. The network will then only upscale its internal feature representation of the image to the same size as the high-resolution image at the end of the network as shown in Figure 2.11. The key enabler for most post-upsampling networks, is an upsampling operation called *pixel shuffling* [91]. Although it is not a common problem, a limitation of pixel shuffling is that the upscaling factor must be the same in the vertical and horizontal direction.

2.8.1 Common Components

Although there exists many different super resolution networks, there are a few components that are often present. Here we briefly cover some frequently used network components, though we will mainly focus on those that have been used in this thesis.

Convolutional Layer

The convolutional layer is most commonly associated with the Convolutional Neural Networks (CNNs) and the layers consist of multiple filters. A filter in this setting is a collection of 2D matrices called kernels with weights to be learned [93]. In a convolution layer, the kernels are slided along an input matrix to produce an output consisting of the sum of the element-wise product between the kernel and

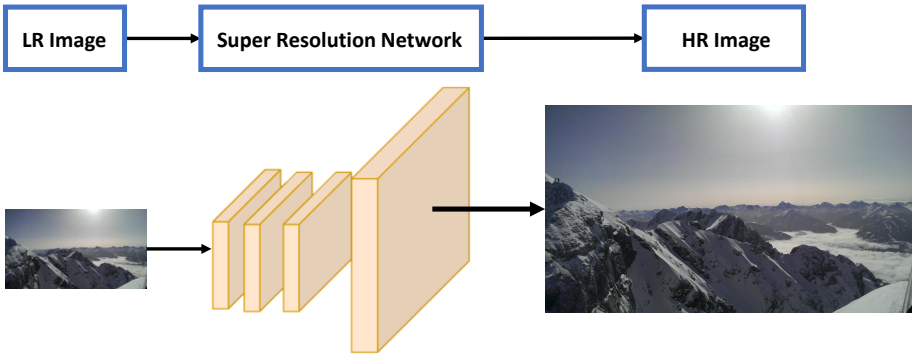


Figure 2.11: Example of the pipeline for a post-upsampling based super resolution network, where the low-resolution (LR) image is used directly as the input, while the final high-resolution (HR) image is created by only upscaling at the end of the network.

the matrix as shown in Figure 2.12. Technically this operation is called 2D cross-correlation, but in the deep learning domain this term is often used interchangeably with the convolution operation, as they only differ by a 180° rotation of the kernel [94]. Since the weights of the kernel are parameters to be learned, the distinction between these operations are for practical purposes not relevant.

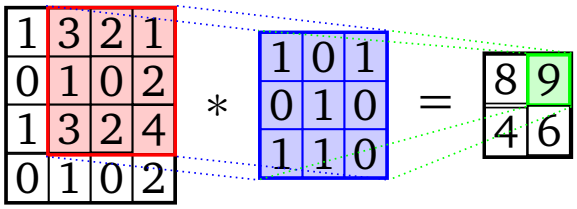


Figure 2.12: Illustration of a convolution operation between a 4x4 input matrix and a 3x3 kernel.

Transpose Convolutional Layer

The transpose convolutional layer is (usually) used in order to upsample an input to a higher spatial dimension and are commonly found in super resolution network architectures [95]. Transposed convolution is sometimes also referred to as deconvolution, although this is an inaccurate use of terminology, since transposed convolution is not an inverse convolution operation [94]. Transposed convolution layer differs from classical interpolation-based upsampling methods, like for instance bicubic interpolation, in that the upsampling parameters are trainable. This is enabled by leveraging kernels with learnable weights similar to the convolutional layer [95]. An example of the transpose convolution operation can be seen in Figure 2.13, where a 3x3 kernel is rotated 180° and slided along an

input matrix to generate an output matrix with a higher spatial dimensionality. In the figure, the kernel is moved only a single row/column at a time along the input, which is called a (1,1) stride, while a higher stride corresponds to moving the kernel multiple rows or columns at a time [94].

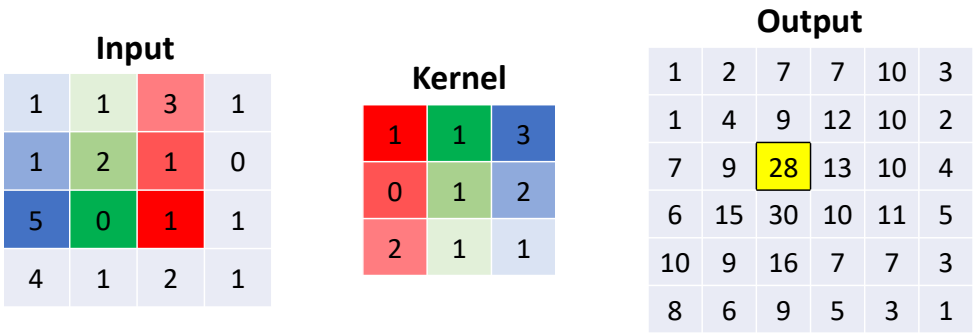


Figure 2.13: This example shows a transpose convolution operation between a 4x4 input matrix and a 3x3 kernel using a (1,1) stride. The highlighted cell in the output matrix is computed as the sum of the element-wise product between the color coded entries in the input matrix and the kernel. Here there have been implicitly added zeros along the boundary of the input matrix to calculate the output in the regions, where there is only partial overlap between the input and the kernel, which is called zero padding [94].

Pooling Layer

A pooling layer is used to shrink the size of the input, while attempting to preserve the most important information. This is done to reduce the number of parameters in the network and the computational requirements [93]. The most used pooling layers are the max and average pooling layers. In a max pooling layer, the input is split into patches and then the maximum value within each patch is extracted [96]. Similarly, in an average pooling layer the average of each patch is extracted.

Fully Connected Layer

In a fully connected layer, each node in one layers is connected to every node in the next layer as shown in Figure 2.14. One of the major challenges with this layer, is that the computational complexity grows quickly with the number of inputs and outputs of the layer [96].

Activation Layer

In an activation layer, a nonlinear function called an activation function is applied to each element in the input. One of the most widely used activation functions is

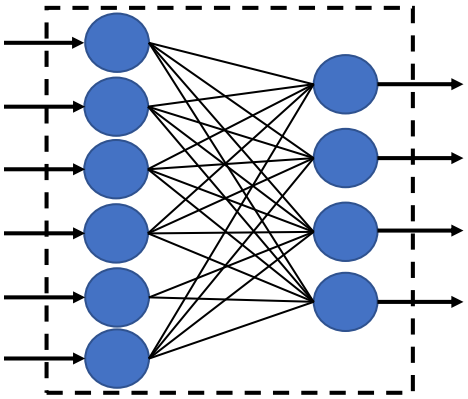


Figure 2.14: A fully connected layer where each node in the previous layer is connected to every node in the next layer.

the Rectified Linear Unit (ReLu) [97], which is given as

$$f(x) = max(0, x) \tag{2.16}$$

for an input x. Two of the major advantages of ReLu are that it is both computationally inexpensive and that it can mitigate a common challenge with training deep network architecture known as the vanishing gradient problem, where the early layers in the network are prevented from training effectively [98]. However, one problem with ReLu is that it can become "stuck" in a state where it only outputs zeros if the input is always negative, which is called the dying ReLu problem [99]. There does exist variations of ReLu such as the *leaky ReLu*, which seeks to alleviate this issue by modifying the gradient to be slightly positive for negative input values [100].

2.8.2 Training

Training an ANN like a super resolution network successfully, is known to be challenging [101]. Knowledge about the training procedure and the existing techniques is therefore essential in order to able to train a neural network effectively. We will therefore go through some of the commonly used methods to train a neural network.

Optimization

One of the most common methods to train a neural network, is to iteratively optimize a predefined cost function on a training set by incrementally adjusting the weights in the network. An algorithm that is responsible for the weight adjustments is called an optimizer and they are usually based on some variation of the (stochastic) gradient decent algorithm [96]. A very popular gradient decent based

optimization algorithm for deep neural networks, is the *Adam optimizer*, since it often performs relatively well without much parameter tuning [102].

Regularization

A commonly faced challenge when training a deep neural network architecture, is that the model can overfit to the training set, so that it fails to generalize to new data. Multiple techniques have therefore been proposed in order to alleviate this problem, which are called regularization methods.

One way to increase the generalization performance of the network is to acquire more training data. However, collecting more training data can be both time consuming and expensive. A cheaper and well-known regularization method, is to artificially increase the training set by modifying the training data using a realistic set of transformations, e.g. randomly rotating or flipping an image, which is called *data augmentation* [96].

Before training a neural network, we will almost always split the dataset into three different subsets, which are called the training set, the validation set and the test set [96]. The test set is as the name suggests, only used to assess the generalization performance of the final network and must therefore be kept completely out of the training phase. Meanwhile, the training set is what the network is training on. However, in order to estimate when the network is starting to overfit to the training set, we can periodically evaluate the network on the validation set. If the performance on the validation set is starting to decrease, then it is a typical indication that the network is starting to overfit to the training data. Training past this point will often result in worse generalization, so a commonly used method is to stop the training process when the performance on the validation set is degrading, which is a regularization method called *early stopping* [96].

An often-used regularization method for deep network architectures, is to randomly disable some of the nodes in the network during training. This method is called *dropout* and the proportion of nodes that are disabled at the same time is called the dropout rate. The main idea behind dropout it is that it forces the nodes in the network to become less reliant on other specific nodes. This can result in the individual nodes becoming more robust to bad inputs and it can be an effective tool to reduce overfitting [103].

A challenge with training deep network architectures, is that the distribution of the inputs to a layer may change significantly when the weights in the network are updated, which can destabilize the training process. A popular approach to mitigate this problem is to apply *batch normalization* [104], where the input to each layer in the network is normalized using the mean and variance of the current batch. A batch in this setting is a set of samples from the training set that are used for a single update of the weights in the network and the number of samples in a batch is called the batch size. Empirical results have also shown that batch normalization can introduce regularization effects, which can result in improved generalization [104].

Chapter 3

Methodology

This chapter covers the approach that was taken to solve the research problems presented in Section 1.2. The chapter is organized as follows: First a new lidar-based place recognition algorithm is presented in Sections 3.1 and 3.2. Then in Section 3.3, a pre-existing lidar super resolution pipeline is covered. Sections 3.4 to 3.6 goes through how the pre-existing lidar super resolution pipeline was leveraged to super-resolve range, intensity and ambient lidar images. Section 3.7 covers a new approach to generate lidar images with an arbitrary resolution, that can be used to train a lidar super resolution network. Section 3.8 goes through the development process of a sensor rig that has been built as part of the thesis, while Section 3.9 covers the data collection using the sensor rig. Finally, Sections 3.10 to 3.12 goes the experimental procedure to test the methods presented in the chapter.

3.1 Lidar-Visual Place Recognition

The first focus topic for the thesis presented in Section 1.2 was to use both the intensity and ambient images from a lidar, to attempt to improve robotic place recognition. Therefore, in Section 3.2, a new place recognition pipeline that leverages both the intensity and the ambient image from a lidar, is presented. However, before we go through the pipeline, we will cover some of the relevant challenges of using these images for place recognition and how the problem to be solved was restricted.

As we saw in Figure 2.2, lidar ambient images are highly dependent on the lighting conditions of the scene within the near-infrared spectrum. Using ambient images for place recognition therefore poses a challenge in terms of potentially having a high degree of noise and changing illumination conditions when revisiting a location. Meanwhile, lidar intensity image is for the most part unaffected by changing illumination conditions, but as we saw in Figure 2.3, it can be challenging to distinguish structures that are far from the lidar. This can potentially make it difficult to differentiate relatively open areas. As we have also seen, both images have a relatively low resolution and quality when compared to regular

cameras. Since the ambient image is usually dominated by noise when the lidar is in low-light environments, we limit the place recognition problem to outdoor environments during daytime, which still poses a challenge in terms of changing illumination conditions and shadows.

3.2 Place Recognition Pipeline

A flowchart showing the main steps in the proposed place recognition algorithm is shown in Figure 3.1, and in Sections 3.2.1 to 3.2.5 we cover each step of the pipeline. The algorithm uses a BoVW-based approach, which as mentioned in Section 2.4.2, is widely used in visual place recognition and hence draws inspiration from prior BoVW-based methods [35, 40, 43]. In particular, the algorithm is influenced by the lidar-VPR algorithm by Shan. et al. [43], which was mentioned in Section 2.4.3. However, their method did not leverage ambient lidar images, which differs from the algorithm that is presented here. Another important note, is that the algorithm presented here is a purely appearance-based place recognition algorithm, which does not depend on any full-scale SLAM pipeline. The reason for this choice is a combination of two factors. The first factor is that it is meant as a proof-of-concept for combining intensity and ambient images for place recognition. The second factor is that this enables the algorithm to be used as part of a re-localization fallback, if it is later integrated into a full SLAM pipeline.

3.2.1 Initialization

The proposed place recognition algorithm is based on a BoVW approach. For this purpose, we use the open-source BoVW library DBoW3 [105], which is a later iteration of the original DBoW library described in Section 2.4.2. Using DBoW3, we create two different databases during the initialization of the algorithm, DB_{Amb} and DB_{Int}, corresponding to the ambient and intensity lidar images. These databases will always be synchronized in the sense that pairs of ambient and intensity images will always be added at the same time. A necessary prerequisite to initialize the databases using DBoW3, is that we have access to a visual vocabulary for each of the databases that has been trained offline. Therefore there have been trained two individual visual vocabularies, one for the ambient images and one for the intensity images, which is addressed in Section 3.10.1.

3.2.2 Feature Detection and Description

Assume now that a new ambient and intensity lidar image pair (I_{Amb}, I_{int}) has been received by the place recognition algorithm. The first step of the algorithm is to detect and extract features from the two images. Here we specifically use the ORB descriptor [35] from the OpenCV computer vision library [106]. This descriptor was chosen due to its properties of being computationally efficient,

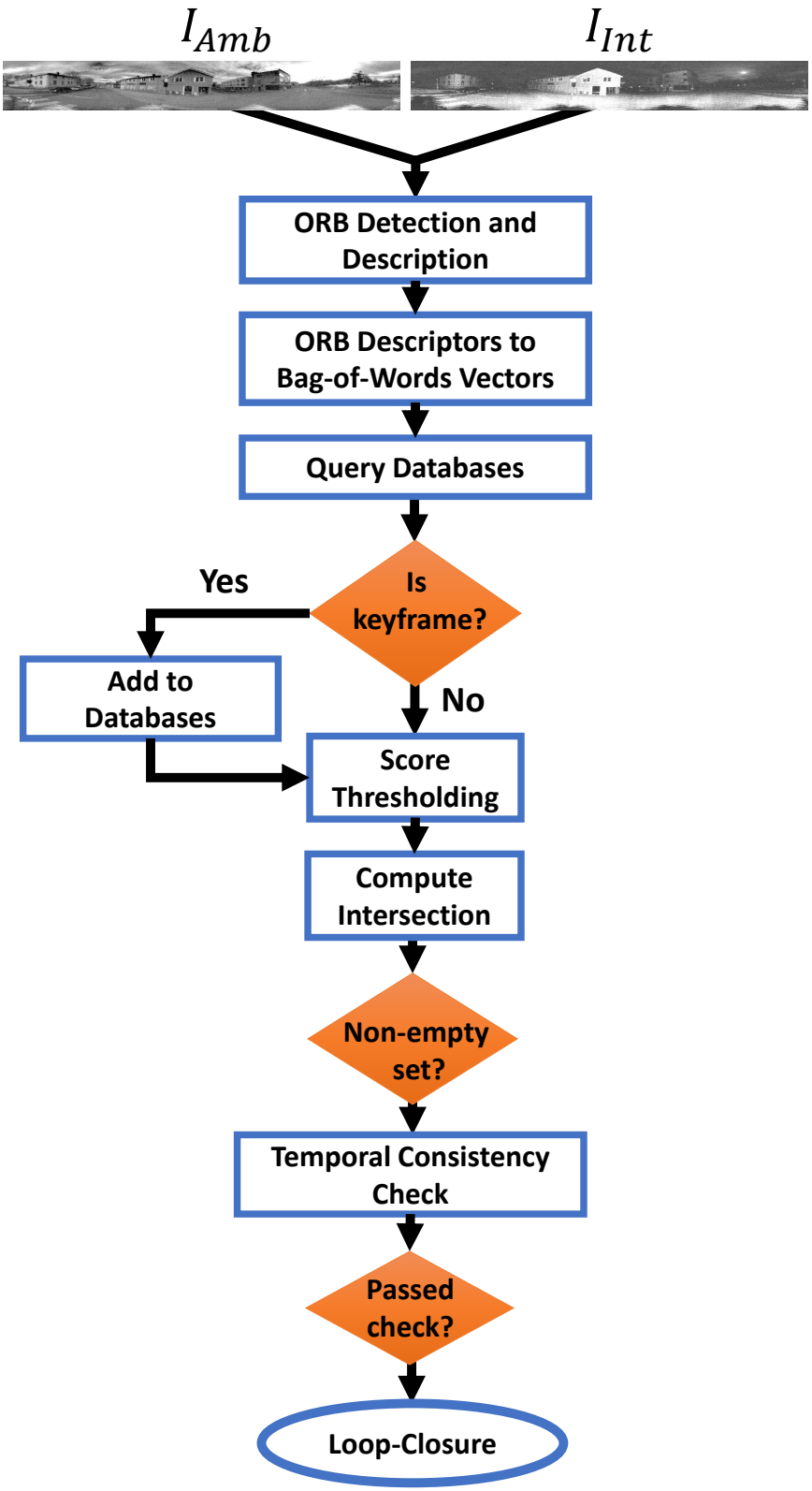


Figure 3.1: Flowchart of the place recognition algorithm.

invariant to in-plane rotations, having partial scale-invariance and that is has been demonstrated to work well for VPR [22].

Using ORB, we detect and extract N_F ORB descriptors from each image. Let $\mathcal{F}_{Amb}$ and $\mathcal{F}_{Int}$ denote the resulting sets of ORB descriptors extracted from I_{Amb} and I_{int} respectively. We then transform I_{Amb} and I_{int} into the DBoW bag-of-words vectors $\mathcal{V}_{Amb}$ and $\mathcal{V}_{Int}$, by using the respective ambient and intensity vocabularies from the initialization described in Section 3.2.1.

3.2.3 Image Retrieval

In the next step we use $\mathcal{V}_{Amb}$ and $\mathcal{V}_{Int}$ to perform a query to their respective database to retrieve the N_s highest scoring images, that are at least T_{min} seconds back in time. We then extract pairs of ambient and intensity images by finding the intersection between IDs of the retrieved ambient and intensity images. As mentioned in Section 3.2.1, the databases DB_{Amb} and DB_{Int} are always synchronized, which implies that if the image IDs are the same, then the images are from the same time instance. The ID intersection therefore works as a voting mechanism that reduces the likelihood of false positives.

3.2.4 Score Thresholding

To further reduce the number of false positives, we introduce two thresholds α_A and α_I, for the ambient and intensity images respectively. If the score for either the ambient image or the intensity image in a retrieved image pair is lower than their respective threshold, then it is not considered as a loop-closure candidate. If there are any image pairs remaining in the intersection set after the score filtering, we extract the pair that has the highest weighted score as a loop-closure candidate. In the current implementation, the ambient and intensity scores are weighted evenly, but other weighting functions are possible. Finally, irrespective of whether we found a loop-closure candidate pair, we check if there has been at least τ seconds since we last added an intensity-ambient image pair to the databases. If this condition is fulfilled, then the intensity-ambient image pair is referred to as a keyframe, where we add the ambient and intensity image to the databases DB_{Amb} and DB_{Int} respectively.

3.2.5 Loop-Closure Verification

If a loop-closure candidate was found in the preceding step, then we need to further validate it to limit the number of false positives. If the loop-closure candidate is indeed a true positive, then it is likely that the following frames will also have a large score for the same image. A voting-based temporal consistency check is therefore used, where if the M next ambient and intensity pairs also have a score more than α_A and α_I for the same image, then it is classified as a loop-closure.

3.3 Lidar Super Resolution

We now shift to the second focus topic of the thesis described in Section 1.2, which was to explore applying neural networks to increase the apparent resolution of the range, intensity and ambient images from a real lidar. As we saw in Figure 1.2, the main challenge with the lidar image resolution, is the vertical image resolution, since this is restricted by the channel-number of the given lidar. Meanwhile, the horizontal resolution is usually only restricted by the sampling frequency. The focus was therefore narrowed down to attempt to increase the apparent vertical lidar image resolution, while the horizontal resolution should be preserved.

The method to be presented builds upon the work by Shan et al. in [79] where they present a lidar super resolution pipeline, which was mentioned in Section 2.7.1. We will therefore start off in Section 3.3.1 by covering the key components of their method in greater detail, and then in Sections 3.4 to 3.7 we go through the approach that was taken in this thesis.

3.3.1 Simulation-Based Lidar Super Resolution

In the lidar super resolution paper by Shan et al. [79], they seek to increase the apparent channel-number of a given lidar with respect to the range information. To achieve this, Shan et al. uses a simulation-based deep learning approach. We will here start off by explaining their data generation process, since this will be necessary to understand how the super resolution pipeline works, and how it differs from the work in this thesis.

Data Generation

In the method proposed by Shan et al. [79], they assume that the lidar we want to increase the resolution of is known in advance, so that the lidar's channel-number C, vertical field of view F_v and horizontal field of view F_H can be identified. They also assume that it is known by which factor X we seek to increase the lidar resolution with. Given this information, they simulate a lidar with the same characteristics as the original lidar, except that they set the channel-number to $C \times X$. They then use spherical projection to convert the lidar point cloud into a 2D image, which results in an image similar to what we have seen up until now as the range image from the Ouster lidar. The range image is then decimated by extracting every X-th row from the image, which simulates the range image that would be perceived by the original lidar as shown in Figure 3.2. The pre-decimated and post-decimated range image pair (I_{HR}, I_{LR}) will then constitute a high-resolution and low-resolution image pair that can be used to train a neural network.

Network Architecture

After extracting the high and low-resolution image pairs, Shan et al. [79] takes inspiration from the SISR problem and propose to use a neural network that

(a) High-resolution range image with $C \times X$ channels.

(b) Low-resolution range image with C channels

Figure 3.2: Example of how the high-resolution range image (a) and low-resolution image (b) are generated. Initially a lidar with $C \times X$ channels is simulated to generate the high-resolution range image. Then every X-th row is extracted to generate the low-resolution image. In this illustration we have that $C = 64$ and $X = 2$.

can learn a mapping from the low-resolution range images to the high-resolution range images. Since the dimensions of the network is dependent on both the desired upscaling factor and the size of original low-resolution image, we will here mainly give an outline of the network architecture and focus on the most important parts of it. For the interested reader, the layers in the network are provided in Appendix A.1. The network can also be found in the original paper [79] or in the code by Shan et al., which is openly available on GitHub [107].

An overview of the network architecture proposed by Shan et al. is shown in Figure 3.3. The network architecture can mainly be broken into two stages. First

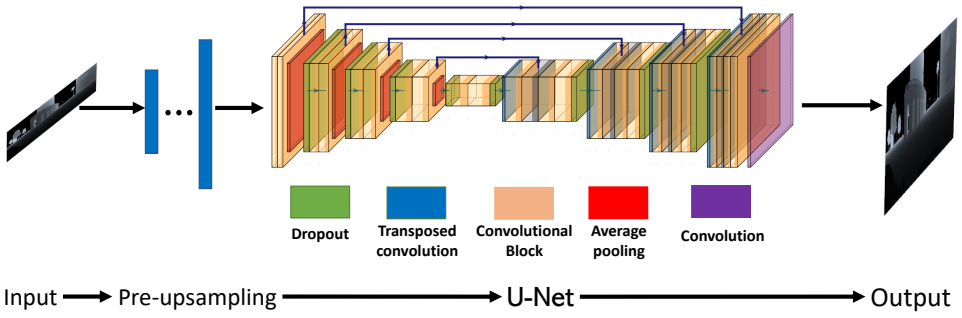


Figure 3.3: Overview of the range image super resolution architecture proposed by Shan et al. Figure adapted from [79].

the low-resolution image is pre-upsampled to the same resolution as the high-resolution image. The upsampling is performed with N transposed convolution layers, where N is given by

$$N = \frac{\log(X)}{\log(2)} \tag{3.1}$$

and X is the desired vertical upscaling factor.

The second stage of the network is a modification of a convolutional neural network architecture called U-Net, which was originally developed for biomedical image segmentation [108]. The U-Net architecture consists of two different parts, which are called the contraction path and the expansion path as illustrated in Figure 3.4. The contraction path in the modified U-Net by Shan et al. [79] is made

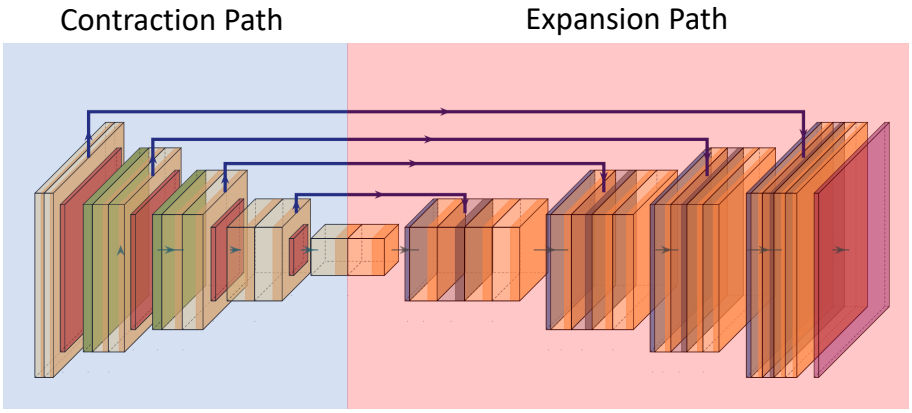


Figure 3.4: The contraction path (blue) and the expansion path (red) in a U-Net based network architecture. The aim of the contraction path is to gradually reduce the spatial dimension while extracting an increasing amount of features, while the expansion path will try to recreate an image from the features extracted in the contraction path [108].

up of a series of convolutional blocks and average pooling layers, which are used to convert the image into a compact feature representation of low spatial dimensionality. The convolutional block in this setting consists of a convolutional layer, a batch normalization layer and a ReLu activation layer, two times in succession as shown in Figure 3.5.

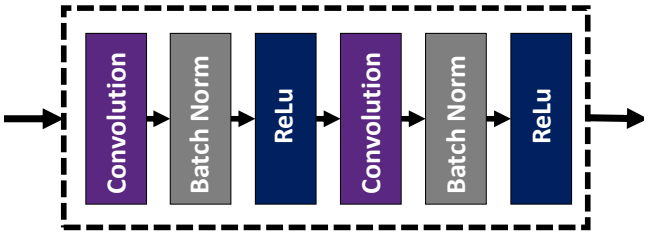


Figure 3.5: A convolution block in the lidar super resolution network by Shan et al. [79] consisting of a convolutional layer, batch normalization and a ReLu activation layer, two times in succession.

The expansion path of U-Net-based network by Shan et al. [79] will try to recreate an image with the same dimensions as the pre-upsampled image, based on the features extracted in the contraction path. This operation is performed by a series of convolutional blocks together with transpose convolutional layers, which

gradually increases the spatial dimension. The final output of the expansion path is fed to a single convolutional layer, which generates a high-resolution range image.

The most important modification Shan et al. have made to the original U-Net architecture, is to add multiple dropout layers inside the network as can be seen in Figure 3.3. However, before explaining the motivation for adding the dropout layers, we will cover a core problem addressed in their paper [79].

Noise filtering

One of the main challenges Shan et al. seeks to address in [79] is best illustrated as shown in Figure 3.6. When a lidar scans along an edge boundary, there will be hard discontinues in the resulting range image. If this range image is fed into a neural network containing convolutional layers, then these layers will result in smoothing along the edge boundaries. If an upscaled range image from the super resolution network is naively projected to a 3D point cloud, then it will likely contain point predictions that are "floating" in the air along the object boundaries, due to these smoothing effects.

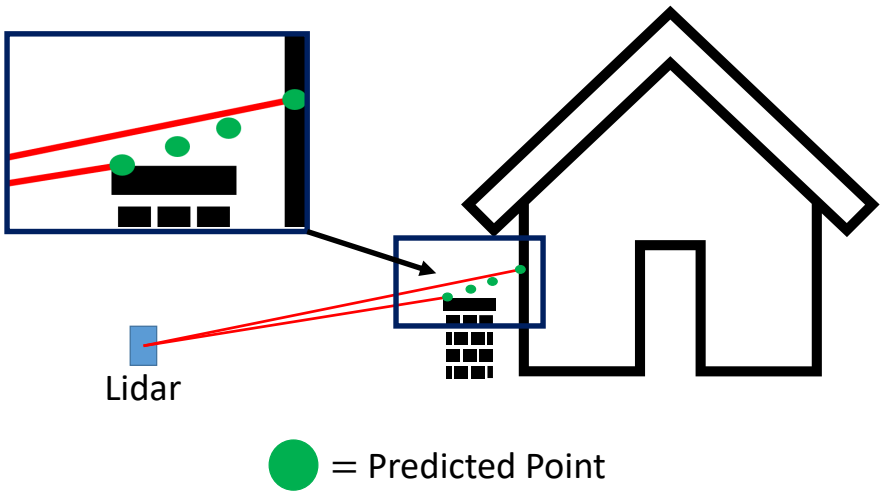


Figure 3.6: Illustration of the point smoothing problem. When convolutional operations are applied in the neural network, the resulting range image will be smoothed out along edge boundaries. The resulting upscaled point cloud will as a result contain points which "float" in the air as shown. Image adapted from [79].

The solution proposed by Shan et al. [79] to alleviate this issue, is to leverage the dropout layers that were added to the U-Net architecture, as we saw in Figure 3.3. The reason for the addition of dropout, is that this enables the network to learn an underlying uncertainty distribution for its range estimates [109]. By keeping the dropout layers in the network during inference, the network will make

different predictions, even when the same image is fed to the network. This makes it possible to assess the network's uncertainty for each individual pixel prediction. The way this is done is as follows:

Assume that the network has made N predictions on the same image. Let $x_{i,k}$ be the i-th prediction of the range corresponding to pixel k and let $\bar{x}_k$ be the average predicted pixel value for the N predictions, which is found simply by

$$\bar{x}_k = \frac{1}{N}\sum_{j=1}^{N} x_{j,k}. \tag{3.2}$$

The network's uncertainty estimate for the given prediction is calculated as the standard deviation of the prediction as

$$\sigma_k = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\bar{x}_k - x_{j,k})}, \tag{3.3}$$

where σ_k is the predicted standard deviation for the range of pixel k.

Shan et al. now propose to remove the points with the highest uncertainty, where the main premise is that points that lie on smoothed edge boundaries are likely to have a high variance [79]. The filtering works by applying a thresholding scheme, where points that do not fulfill the condition

$$\sigma_k < \alpha\bar{x}_k, \tag{3.4}$$

are removed and where α is a parameter to be tuned. The reason for why they scale with the mean $\bar{x}_k$, is that the point prediction variances will usually increase with range, so using an adaptive threshold will be more amendable for points at different ranges [79]. Finally, the super-resolved point cloud is generated by projecting the points from the 2D image back to 3D space.

3.4 Lidar Super Resolution - Problem Specification

As described in Section 3.3.1, the lidar super resolution network by Shan et al. [79] was only trained on lidar range images generated by computer simulation. A limitation with this approach, is that it does not take other potential lidar raster bands, such as intensity and ambient information, into account. In this thesis it was therefore decided to investigate if it would be possible to extend the method by Shan et. al to these additional lidar raster bands.

A challenge with this, is that simulating intensity and ambient data realistically is substantially harder than simulating range data [110]. Here the problem was therefore approached in a different manner, where instead of using computer simulation to generate the training data, it was decided to use a real lidar. Specifically, this was an OS0-128 lidar, which is a lidar with 128 channels by Ouster[1]. However, this has a few limitations that we need to address.

[1]https://ouster.com/products/scanning-lidar/os0-sensor/

3.4.1 Real Lidar Data

A limitation of using a real lidar for the super resolution problem compared to simulation, is that we are restricted by the number of channels available by the lidar. Yet, with a lidar with 128 channels available, we can simulate a lidar with a lower number of channels. An assumption for this to be possible, is that the number of channels is the only difference between the lidars, so the vertical and horizontal field of view has to remain the same. We will therefore for the moment restrict the problem to upsampling to a maximum of 128 channels, where we specifically focus on upsampling from 32 and 64 channels to 128 channels, since these are the two other variations of the OS0 lidar series [2]. A potential solution to how this restriction could be lifted is addressed later in Section 3.7.

Since there to the author's knowledge does not exist any prior attempts on increasing the resolution of multiple lidar raster bands, a general outline of the path to the final solution is given here. A natural question to ask is if it is possible to use a pre-existing super resolution network intended for RGB images and change the individual RGB channels with the raster bands of the lidar. A significant challenge with this approach, is that the pixels values in the range, intensity and ambient image can be very different. As an example, the range image may contain "holes" due to the lidar being unable to resolve the depth. Meanwhile, the intensity and ambient images may be non-zero in this region, since the lidar still outputs the measured ambient and intensity, despite being unable to determine the range. These pixel inconsistencies could potentially make it challenging for a neural network to converge, so combining the upscaling of range, intensity and ambient images into single network architecture was not pursued.

The method proposed here to extend the lidar super resolution problem to intensity and ambient data, is to use three different super resolution networks to upscale each of the individual range, ambient and intensity images individually as illustrated in Figure 3.7. In Sections 3.5 and 3.6, considerations about these networks are addressed, where we start with the range image super resolution network.

3.5 Range Image Super Resolution

To super-resolve the range image, the selected solution was to build upon the lidar super resolution pipeline by Shan et al. [79]. As explained in Section 3.3.1, their pipeline leverages a simulation-based approach, where the lidar point cloud is first projected to a 2D range image. Since the Ouster lidar outputs the range image directly, this initial step is no longer necessary and is therefore skipped.

3.5.1 Range Image Normalization

In the pipeline by Shan et al. [79], the range images are expected to be normalized by the maximum range of the lidar, before they are fed into the rest of the network.

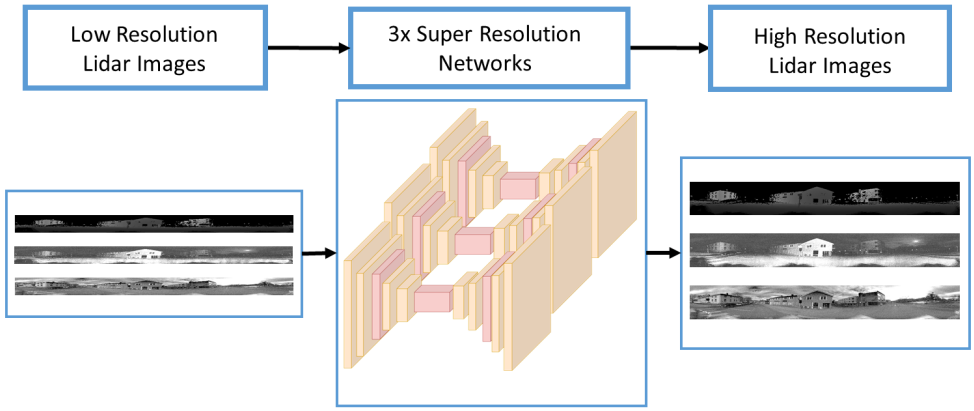


Figure 3.7: Overview of the super resolution pipeline for the intensity, range and ambient images from an Ouster lidar. Each of the three low-resolution intensity, ambient and range images are fed into a separate super resolution network to produce their high-resolution counterpart.

To make the range images from the Ouster lidar compatible with this requirement, we need to go into a few technical details. The range images from the Ouster lidar are discretized with 4 mm resolution into 16-bit images. This results in a maximum possible range of

$$(2^{16}-1)*4*10^{-3}\text{m} \approx 262\text{m}, \tag{3.5}$$

but the specified max range of the OS-128 is only 50 meters [2]. Normalizing by the maximum possible distance for the range image is therefore not a suitable approach, since this will result in improper data normalization. However, even though the specified max range is 50 meters, the lidar image may contain values that are higher than this value. To account for this, we convert the measurements in the range image above 50 meters to 0 m, to indicate that the measurements are invalid. Finally, the image is normalized by the specified max range of the lidar of 50 meters, before it is fed to the lidar super resolution network. Note that to recover the super-resolved range images from the super resolution network, the images have to be rescaled to the original range.

3.5.2 Range Image to Point Cloud

Even though we are training and performing inference on the range image, it is considerably easier to qualitatively evaluate the super-resolved range image by converting it into a 3D point cloud. Here we will therefore go through the process of how this conversion was done.

The coordinate system of the Ouster lidar [111] is shown in Figure 3.8, where we have defined the azimuth angle α and elevation angle β of the lidar. An important thing to note is that the Ouster lidar rotates in the *clockwise* direction around

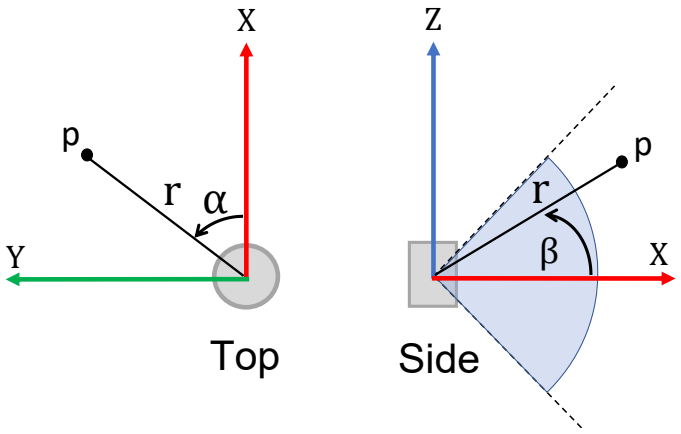


Figure 3.8: Illustration of the lidar coordinate system. The lidar's field of view is indicated in blue. The X-axis points in the forward-facing direction of the sensor while the Z axis points towards the top of the sensor. The indicated angles α and β corresponds to the lidar's azimuth and elevation angles respectively, while r is the range to an arbitrary point p.

the positive Z-axis, which corresponds to a negative angular velocity for α [111]. From Figure 3.8 we also see that the Cartesian coordinates of the point p can be found as

$$x = r\cos\alpha\cos\beta \tag{3.6a}$$
$$y = r\sin\alpha\cos\beta \tag{3.6b}$$
$$z = r\sin\beta. \tag{3.6c}$$

Assume now that we are given a range image I_R with dimensions $m \times n$. Let u and v represent the column and row index of I_R respectively, as shown in Figure 3.9. Note that we in the derivation here assume that the image coordinates u and v are 0-indexed, so that $u \in [0, 1, \ldots, n-1]$ and $v \in [0, 1, \ldots, m-1]$. The

Figure 3.9: Illustration of the range image coordinate system.

range image from an Ouster lidar is centered in the lidar coordinate system's positive x-axis [111]. The connection between the azimuth angle α and elevation angle β for the lidar can therefore be illustrated as shown in Figure 3.10. Note that the coordinate vector for α is pointing leftwards, while the coordinate vector

for u is pointing rightwards, since the image is generated by rotating the lidar in the clockwise direction.

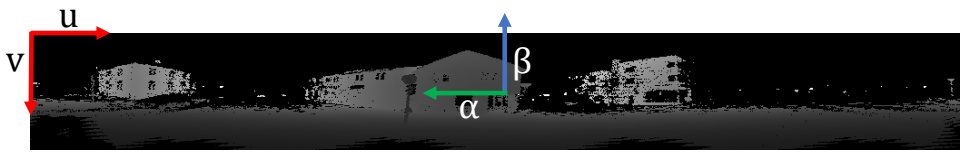


Figure 3.10: Illustration of how the pixel coordinates (u, v) in the image coordinate system relates to the azimuth and elevation angle pair (α, β) for the Ouster lidar.

To be able to convert the range image into a point cloud, we need to know the vertical field of view of the lidar. Let FOV_{up} and FOV_{down} be the maximum and minimum angle for β respectively as shown in Figure 3.11 and define

$$FOV = FOV_{up} - FOV_{down}, \tag{3.7}$$

where FOV is the total field of view of the lidar.

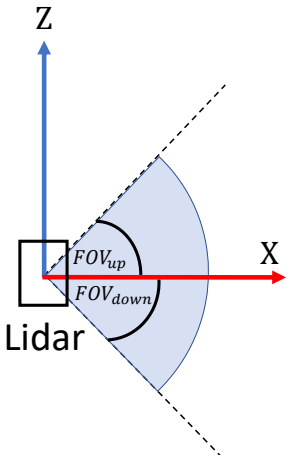


Figure 3.11: Definition of the maximum and minimum vertical field of view of the lidar.

For the next step, we need to know how the lidar beams are distributed over the lidar's vertical field of view, which is also called the beam configuration. In the work presented in this thesis, the OS-128 lidar is specified to have lidar beams that are uniformly distributed over the vertical field of view[2]. This implies that it is designed to have a constant vertical angular resolution, which we will assume is the case for the remainder of the derivation. Since it will be easier to use angles

[2]https://ouster.com/products/scanning-lidar/os0-sensor/

that are aligned with the coordinate system of the range image, we define

$$\hat{\alpha} = \pi - \alpha \tag{3.8a}$$

$$\hat{\beta} = FOV_{up} - \beta, \tag{3.8b}$$

where $\hat{\alpha}$ and $\hat{\beta}$ has been shifted and rotated to be aligned with the image coordinate system as shown in Figure 3.12. Since the range image covers 360° in

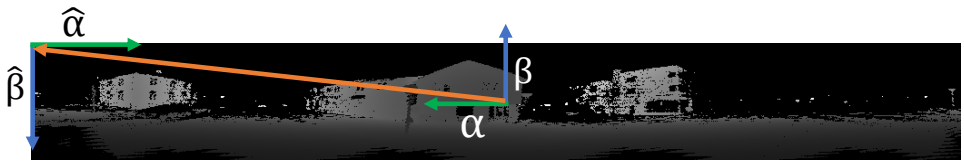


Figure 3.12: Conversion between (α, β) and $(\hat{\alpha}, \hat{\beta})$

the horizontal direction and has a constant horizontal angular resolution, we see that $\alpha \in [-\pi, \pi\rangle$ and hence $\hat{\alpha} \in [0, 2\pi\rangle$. The mapping from the range image's column and row index (u, v) to $(\hat{\alpha}, \hat{\beta})$ can then be found as

$$\hat{\alpha} = \frac{2\pi}{n-1}u \tag{3.9a}$$

$$\hat{\beta} = \frac{FOV}{m-1}v. \tag{3.9b}$$

By combining Equations (3.6), (3.8) and (3.9) we can for a point $(u_i, v_i) \in I_R$ with range value r_i compute its position (x_i, y_i, z_i) in the lidar frame as

$$x_i = r_i \cos\left(\pi - \frac{2\pi}{n-1}u_i\right)\cos\left(FOV_{up} - \frac{FOV}{m-1}v_i\right) \tag{3.10a}$$

$$y_i = r_i \sin\left(\pi - \frac{2\pi}{n-1}u_i\right)\cos\left(FOV_{up} - \frac{FOV}{m-1}v_i\right) \tag{3.10b}$$

$$z_i = r_i \sin\left(FOV_{up} - \frac{FOV}{m-1}v_i\right), \tag{3.10c}$$

which enables us to convert the range image into a point cloud as intended.

3.6 Intensity and Ambient Image Super Resolution

To enhance the resolution of the lidar intensity and ambient images, the initial idea was to explore the use of a post-upsampling super resolution network, since these have in many cases showed good results for the SISR problem [92, 112]. A challenge that was faced with this approach, was as mentioned in Section 2.8, that these networks are usually designed to use pixel shuffling to upsample the images at the end of the network, which requires that the scaling factor is the same

in both the vertical and horizontal direction [91]. However, since the horizontal lidar image resolution is only restricted by the sample frequency of the lidar and not the channel-number, the horizontal resolution is usually not the main issue. Downsampling the lidar images in the horizontal direction and then artificially super-resolving them is also far from optimal, so this initial approach was dropped.

A pre-upsampling network on the other hand, is usually not restricted by this technicality. This comes from the fact that the input dimensions to the network is the same as the output dimensions of the high-resolution image [81]. The final solution was therefore to adopt the super resolution network architecture by Shan et. al. [79], since this is designed to only perform vertical upscaling and explore if this could generalize to the other lidar raster bands.

A key difference here, is that for the range images it is crucial to remove as many outlier points as possible to avoid "floating" points in the point cloud as described in Section 3.3.1. For the intensity and ambient images on the other hand, this is not a concern. Since we for the intensity and ambient images do not need to remove uncertain points, this removes the necessity of performing multiple passes through the network with dropout during inference for point filtering. Dropout can on the other hand be used as an effective tool to mitigate overfitting during training [103]. The approach taken here was therefore to keep the dropout for training and disable it during inference. Also, to normalize the images, we divide the pixel by the maximum ambient and intensity pixel value, which is 65535, since these are 16-bit images that are scaled to use the entire 16-bit range.

3.7 High resolution image generation

One problem with downsampling real lidar images to create low-resolution images, is as mentioned in Section 3.4.1, that it is not possible to generate a dataset with a higher channel-number than what the lidar provides. Super resolution networks trained on such images would therefore also be restricted to this maximum channel-number resolution. Here we specifically address this issue, by proposing a pipeline with the goal of generating lidar images (either intensity, ambient or range) with an arbitrary resolution $m \times n$. The principal idea to achieve this, is to combine multiple point clouds that are captured in close vicinity into a single densified point cloud and leverage that the raw intensity and ambient data is available for each point in an Ouster point cloud. The problem to be solved is the following: Assume that we are given a set of $2N+1$ point clouds that are captured in a small region

$$\mathcal{C} = \{C_{i-N}, \ldots, C_i, \ldots, C_{i+N}\}, \tag{3.11}$$

where C_j corresponds to point cloud j and $\mathcal{C}$ is the total point cloud set. Also assume that an estimate of the relative pose between the point clouds is known and define point cloud C_i as a reference point cloud. The problem is then to generate a lidar image $I_{m\times n}$ at the location of the reference point cloud C_i, based on the set of point clouds $\mathcal{C}$.

An overview of the proposed pipeline that seeks to solve this problem is given in Figure 3.13 and in Sections 3.7.1 to 3.7.5, each of the steps in the pipeline are explained.

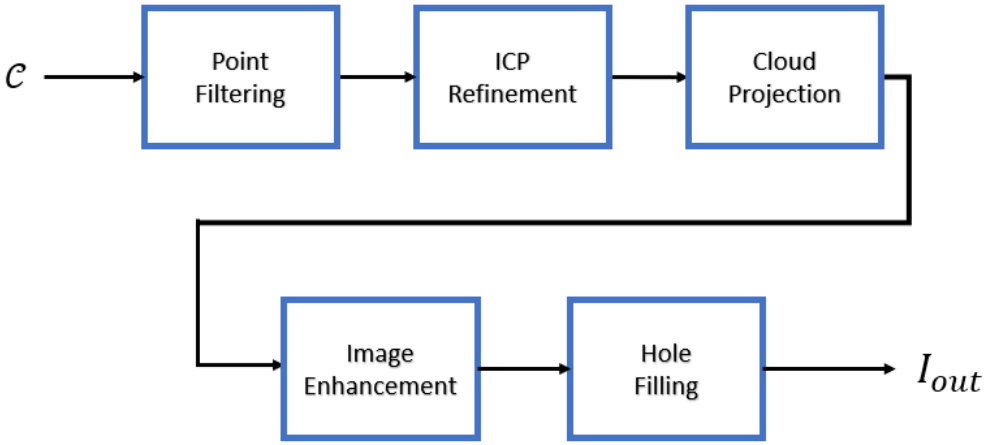


Figure 3.13: Overview of the proposed pipeline to create high-resolution lidar images. A reference lidar cloud C_i and the $2N$ surrounding clouds are extracted to generate a point cloud set $\mathcal{C} = \{C_{i-N}, \ldots, C_i, \ldots, C_{i+N}\}$, where an initial estimate of the relative pose between the clouds is known. In *Point Filtering*, points within a given distance from the center of the point cloud are removed to avoid reflections from e.g. the sensor platform. The filtered clouds are aligned in *ICP Refinement* to create an aggregated point cloud **C**. The aggregated point cloud is projected to an image in *Cloud Projection* and the resulting image is post-processed in *Image Enhancement*. Holes resulting from missing reflections are filled in *Hole Filling* to generate the final range, intensity and ambient images, which are denoted here as I_{out}.

3.7.1 Point Filtering

Since there are usually noisy points close to the lidar, e.g., due to reflections from a person holding the lidar or from the platform the lidar is mounted on, the first step of the pipeline seeks to filter away these noisy points. Here we apply a cube shaped distance filter to each point cloud, where the coordinate system of each cube is aligned with the coordinate system of its respective point cloud. We then select a given side length L for the cube based on tuning and remove all the points located within the cube from the corresponding cloud. Note that a radius based distance filter might be more suitable in some cases, depending on the geometry of the sensor rig used in the given dataset.

3.7.2 ICP Refinement

The second step of the pipeline is to refine the alignment between the point clouds in the set $\mathcal{C}$. Here we use the ICP algorithm [52], since this usually results in an

accurate point cloud alignment, if the initial pose estimate is close enough to the solution and that the problem is sufficiently constrained. All the point clouds in $\mathcal{C}$, except C_i, are aligned with the reference point cloud C_i, using the ICP-based point cloud alignment function *pcregistericp* in MATLAB. After the alignment, we combine all the point clouds in $\mathcal{C}$ into a single aggregated point cloud $\mathbf{C}$ as illustrated in Figure 3.14.

3.7.3 Cloud Projection

After generating the aggregated point cloud $\mathbf{C}$, the next step of the pipeline is to use spherical projection to project $\mathcal{C}$ onto an image of dimensions $m \times n$. Since projecting from a point cloud to a 2D image is essentially the reverse operation of the 2D to 3D projection described in Section 3.5.2, we use the same notation and coordinate systems here. Given the lidar coordinate system defined in Figure 3.8 and Equation (3.6), we see that the spherical coordinates (r, α, β) based on the Cartesian coordinates (x, y, z) of a point p is given as

$$r = \sqrt{x^2 + y^2 + z^2} \tag{3.12a}$$

$$\alpha = \text{atan2}(y, x) \tag{3.12b}$$

$$\beta = \arcsin\left(\frac{z}{r}\right), \tag{3.12c}$$

where $atan2$ is the two-argument arctangent function. To convert the angles α and β into image coordinates (u, v) as in Figure 3.10, we use the same transformed angles $(\hat{\alpha}, \hat{\beta})$ as in Equation (3.8) and based on Equation (3.9), we see that

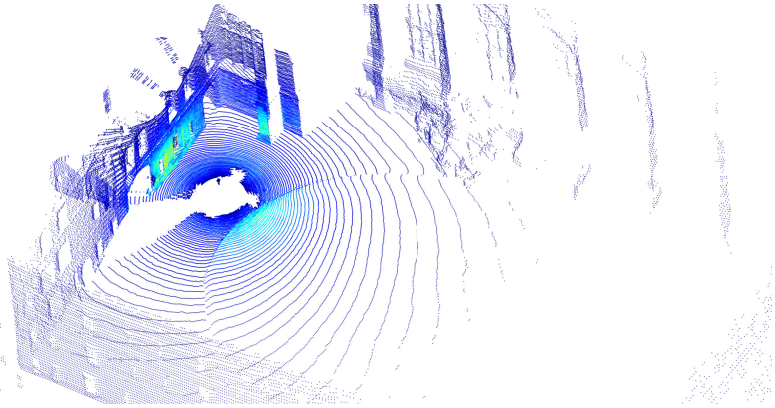
$$u = \text{round}\left(\frac{n-1}{2\pi}\hat{\alpha}\right) \tag{3.13a}$$

$$v = \text{round}\left(\frac{m-1}{FOV}\hat{\beta}\right), \tag{3.13b}$$

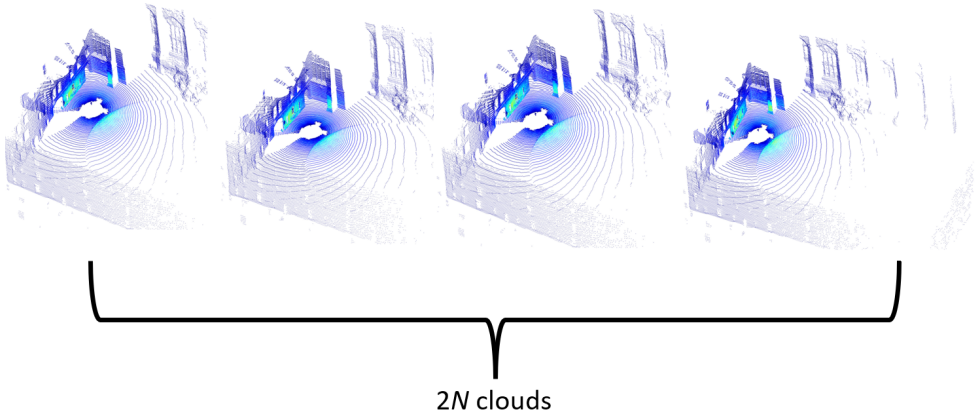
where we also have applied the rounding operator, since image coordinates can only be integer values and therefore needs to be discretized. After the image coordinates of each point in the aggregated point cloud $\mathbf{C}$ have been found, there might be some points that map to the same image coordinates. In this case we choose the median range, intensity and ambient value of the points that map to the same pixel as the representative value for each of the three different images.

3.7.4 Image Enhancement

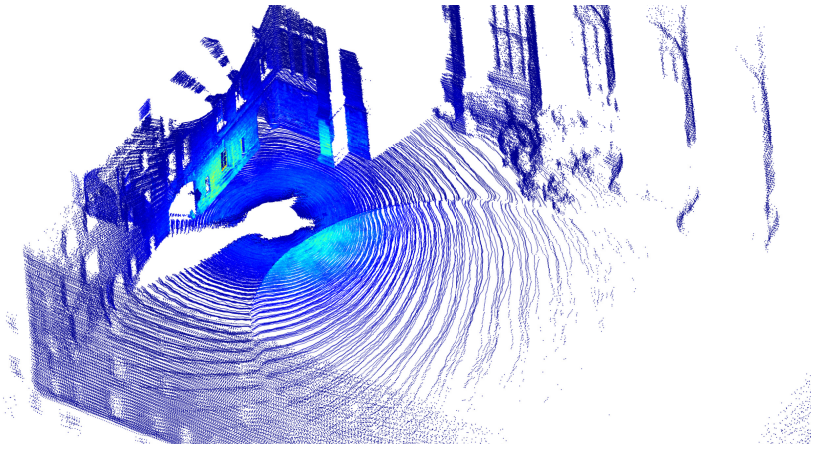
As mentioned in Section 3.7, an Ouster point cloud store the raw intensity, ambient and range value for each point in the point cloud. However, the Ouster intensity and ambient images do not display the raw measurement values, since this is not suited for visualization. This implies that if we want to generate lidar images that look like those by Ouster, we need to apply the same image enhancement methods on the raw image data as they use. This is possible, since the code used by



(a) Single reference point cloud



(b) Surrounding clouds



(c) Aggregated point cloud

Figure 3.14: Illustration of the point cloud aggregation procedure. A point cloud at a given location is selected as a point cloud reference (a). $2N$ surrounding clouds (b) are then extracted and aligned with the reference point cloud using ICP to construct an aggregated point cloud (c). Figures generated based on data from the Newer College Dataset by Zhang et al. [113].

Ouster to generate the lidar images is openly available on GitHub[3], and therefore also the image processing. The image processing consists of applying a non-linear mapping to stretch the raw intensity and ambient data between a range of 0 to 1 and use gamma correction to make the images more suitable for human interpretation. However, since their code is implemented in C++, while the rest of the pipeline presented here is in MATLAB, the image processing was re-implemented in MATLAB to make the data compatible. Since this stage of the pipeline is only meant for visualisation purposes to be able to compare the final images with the lidar images by Ouster, the code specific details in the image processing by Ouster is left out here, but can be found in the aforementioned GitHub repository.

3.7.5 Hole Filling

As a result of using point cloud projection to generate the lidar images, it might occur that none of the points in the point cloud projects to a given pixel in the image. This could be due the fact that there are no objects at the given location, for instance when the lidar is looking towards the sky. Another potential reason is that there are bad reflections from an object at the location, so that the lidar is not able to determine the range. The latter problem usually results in small holes in the image, which this stage in the pipeline tries to alleviate by filling them with plausible values. Note that we do not perform the following hole filling algorithm on the range image, but only on the intensity and ambient image.

The hole filling algorithm used in this pipeline is largely heuristic, where if a pixel does not have any point projected to it, the median of the pixels above and below it is selected, if they contain a valid value. If none of the pixels above or below contains any valid value, the pixel is set to 0. Specifically, this was implemented using the function *fillmissing* in MATLAB, using the filling function *movmedian* with a window length of 3. After the hole filling operation, we have reached the end of the pipeline and we are left with the final range, intensity and ambient images.

3.8 Sensor Rig Development

To be able to test the proposed methods, a requirement is to have access to data. Based on the presented place recognition algorithm and the super resolution networks, we can summarize the data demands as follows:

- Both methods requires ambient images.
- Both methods requires intensity images.
- Lidar super resolution requires range images.
- The place recognition algorithm requires a place verification method.
- The dataset must be large and diverse, so that it can be used to train the super resolution networks and the place recognition visual vocabularies.

[3]https://github.com/ouster-lidar/ouster_example/blob/master/ouster_ros/src/img_node.cpp

Here we will for this purpose go over the development process of a handheld multi-modal sensor rig, that has been built in collaboration with members of the Autonomous Robots Lab (ARL) at NTNU. The new sensor rig named *Mjolnir* can be seen in Figure 3.15 and it is a new iteration of an earlier handheld sensor rig at the ARL. A requirement for the handheld sensor rig, was that it should not only be applicable for data collection for this particular thesis, which is covered in Section 3.9, but also for a variety of future perception applications. The sensor rig therefore incorporates multiple different sensor modalities for robotic perception. Note that the CAD modeling of the sensor rig was not done by the author of this thesis, but the contributions of the author includes the full assembly and testing of the sensor rig, as well as selecting and integrating a RTK GNSS sensor, that could be used to validate the place recognition algorithm. The following sections therefore have a special focus on the work that has been done to integrate the RTK GNSS sensor. A note on the development process, is that as a consequence of long delays in shipping, partly due the COVID-19 pandemic, three of the cameras that were originally planned to be installed, had to be dropped. However, since the rest of the system has been prepared for later integration of the cameras, we will also briefly cover how they can be added.

The sensor rig is presented in the following way. First, Sections 3.8.1 and 3.8.2 covers the hardware on board Mjolnir and how the electronics are connected. Subsequently in Section 3.8.3, some of the design aspects of sensor rig are presented, to explain particular trade-offs that had to be made for the RTK GNSS sensor. Finally, in Section 3.8.4 we go into detail about the RTK GNSS sensor and how it was made compatible with the rest of the system.

3.8.1 Hardware

Here we briefly go over the sensors on board the sensor rig, while a complete list of the hardware for the sensor rig is given in Table 3.1

Table 3.1: Overview of the hardware on the Mjolnir sensor rig.

Type	**Model**
Computer	Intel NUC
Lidar	Ouster OS0-128 Gen 2
Cameras	CamBoard pico flexx ZED 2 Stereo Camera
GNSS Board	simpleRTK2B V1
GNSS Antenna	u-blox GNSS Multiband Antenna
IMU	VN-100
Network Switch	USW Flex Mini

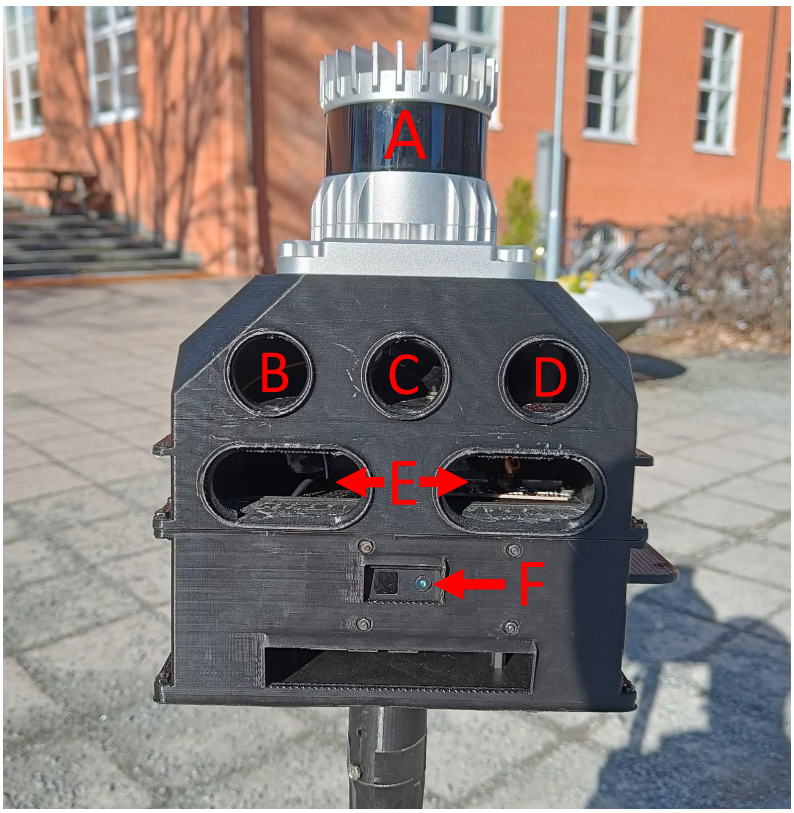


Figure 3.15: Front side of the Mjolnir sensor rig, where we see: (A) an OS0-128 lidar. (B-D) Slots for three mono cameras . (E) Slot for a ZED 2 stereo camera, which was installed later. (F) A CamBoard pico flexx depth camera.

Computer

The computer on board the sensor rig is an Intel NUC[4] running the Ubuntu operating system. The relatively high computing power and storage space (2 TB) of the NUC given its small form factor, makes it a good choice for a handheld system.

Lidar

OS0-128 Gen 2 is a multi-beam lidar by Ouster[5]. The lidar has 128 channels uniformly distributed along it's 90° vertical field of view and has a maximum range of approximately 50 meters.

[4]https://www.intel.com/content/www/us/en/products/details/nuc.html
[5]https://ouster.com/products/scanning-lidar/os0-sensor/

GNSS module

SimpleRTK2B[6] is a standalone board by ArduSimple for GNSS and RTK applications. The simpleRTK2B is based on the u-blox ZED-F9P GNSS module[7], which supports multi-band GNSS and onboard RTK calculations. The ZED-F9P GNSS sensor is compatible with the four major GNSS constellations GPS, GLONASS, Galileo and BeiDou. The simpleRTK2B requires an external antenna for GNSS signal reception, which here is a u-blox GNSS multi-band antenna[8].

Inertial Measurement Unit

VN-100 is an industrial grade Inertial Measurement Unit (IMU) by VectorNav[9]. The VN-100 can output IMU data up to 800 Hz and runs an onboard state estimation algorithm for gyro drift compensation and orientation estimation.

Stereo Camera

ZED 2 is a stereo vision color camera, which can be used for passive depth perception by triangulation[10].

Depth Camera

CamBoard pico flexx[11] is a depth camera by pmdtechnologies. The camera works on the principle of time-of-flight by using active infrared illumination of the surroundings within sensor's field of view.

3.8.2 Electronics topology

An overview of the electronics onboard Mjolnir and how they are connected is given in Figure 3.16. As we see in the figure, the simpleRTK2B, pico flexx, ZED 2 and the VN-100 are all connected directly to the NUC computer by USB for data transmission and power. Since the data rate of the Ouster lidar is substantially higher than the other sensors, it requires a Gigabit Ethernet (GigE) connection. However, since three other cameras are later going to be added, which also require a GigE connection, a network switch was necessary. To make the connection to the switch work, the Ouster lidar and the NUC computer have both been configured with static IP-addresses on the same subnet. This configuration will make it relatively easy to later integrate other camera sensors by assigning them to the same subnet. For power, the Ouster lidar and the network switch have been set

[6] https://www.ardusimple.com/simplertk2b/
[7] https://www.u-blox.com/en/product/zed-f9p-module
[8] https://www.u-blox.com/en/product/ann-mb-series
[9] https://www.vectornav.com/products/detail/vn-100
[10] https://www.stereolabs.com/zed-2/
[11] https://pmdtec.com/picofamily/

up with individual DC-DC converters to convert from the battery's supply voltage to the operating voltage of each device.

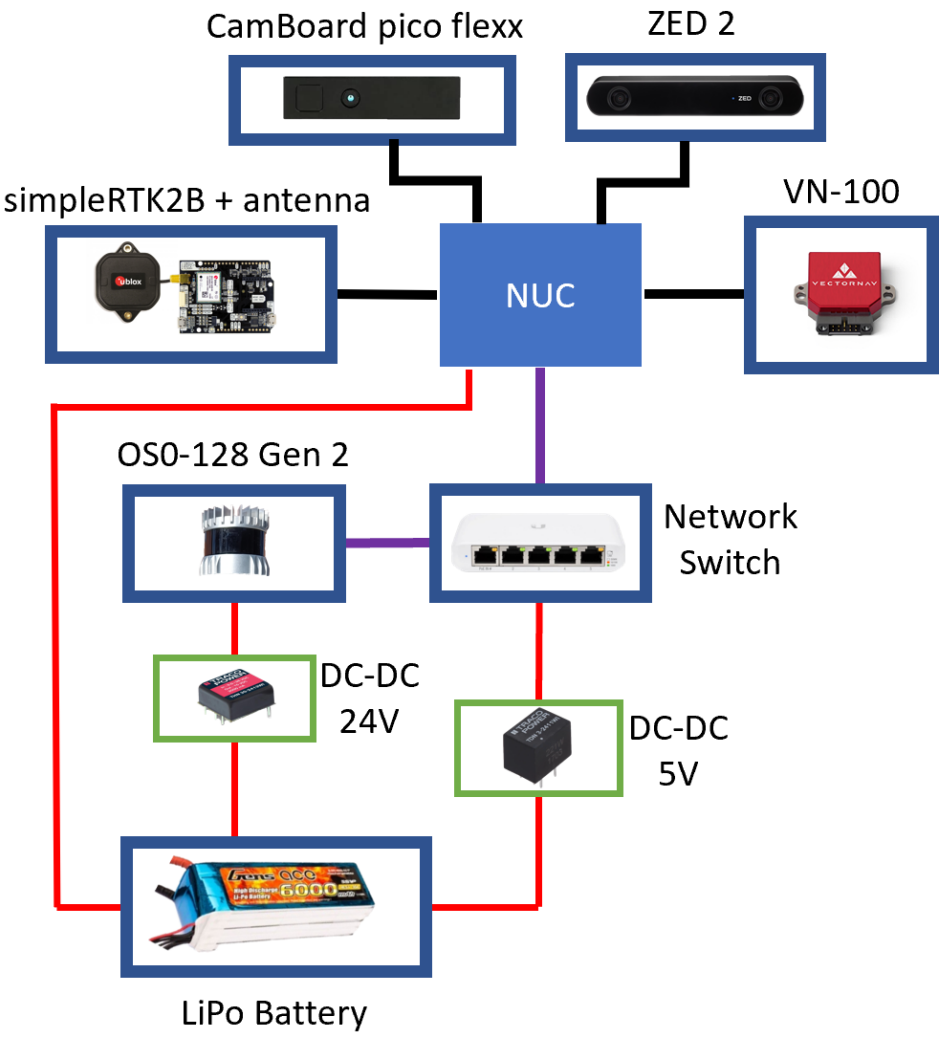


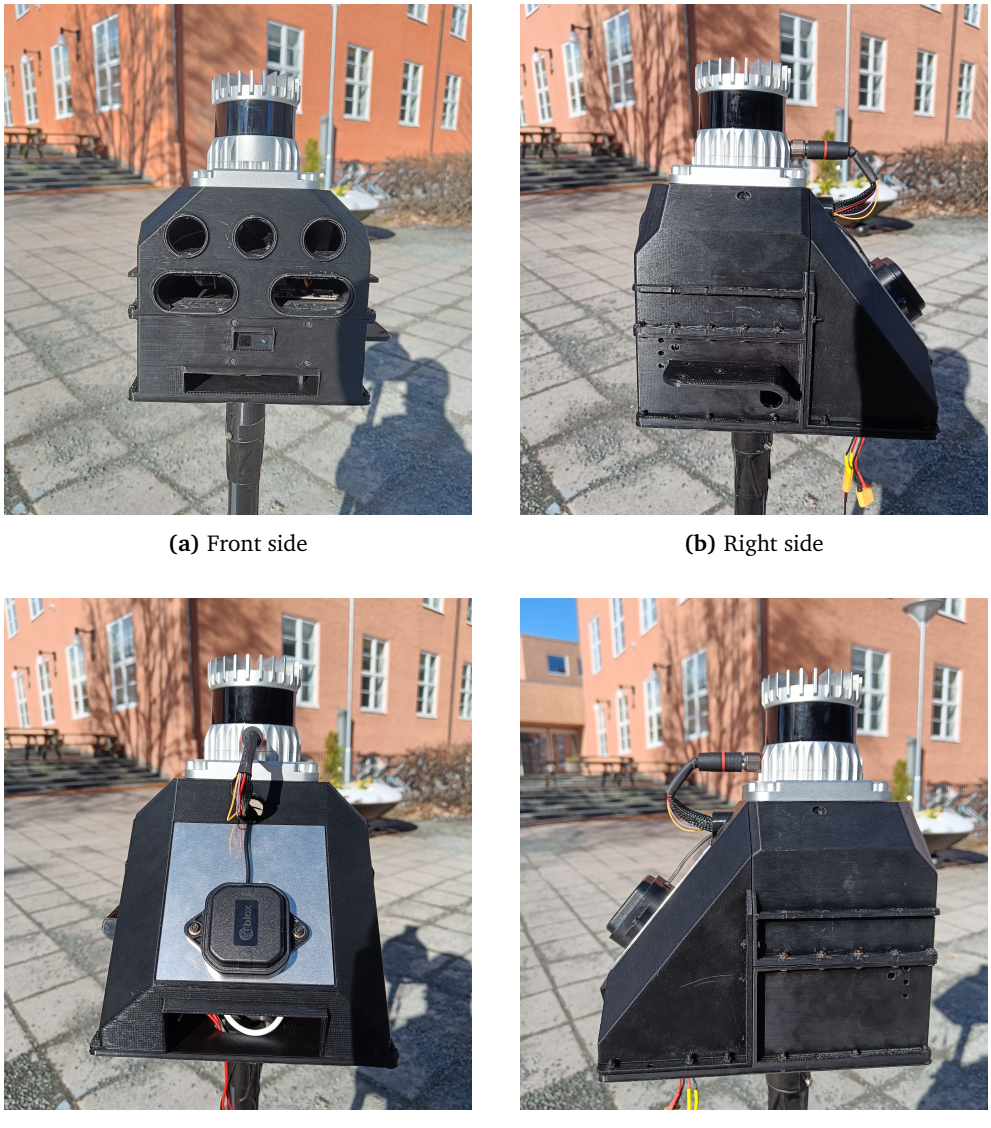
Figure 3.16: Overview of the electronics on board the Mjolnir sensor rig. The black lines represents USB connections, red lines represents wires for power transmission and the purple lines represents Ethernet cable connections.

3.8.3 Exterior Design

The exterior of the Mjolnir sensor rig from all four sides is shown in Figure 3.17. A goal with the design of the sensor rig, was that it should minimize obstructions within the field of view of the lidar. In order to accommodate this, the housing of the sensor rig was designed with angled top corners and an angled back plate, as can be seen in Figure 3.17. On the back side in Figure 3.17c, the GNSS antenna

is seen mounted on top of an aluminium metal plate, which is used as a ground plane for improved signal reception.

Ideally, the GNSS antenna would be mounted in a horizontal rather than angled position to maximize signal reception. Given that one of the goals for the design was minimize obstructions in the field of view of the lidar, a trade-off had to be made for where the GNSS antenna could be placed. Since a large part of the backwards facing direction of the lidar is already blocked by its connection cable, mounting it with the given angled position resulted in only a minimal reduction in the field of view of the lidar.



(a) Front side

(b) Right side

(c) Back side

(d) Left side

Figure 3.17: Exterior of the Mjolnir sensor rig shown from all four sides.

3.8.4 GNSS integration

Here we cover how the simpleRTK2B application board with the ZED-F9P GNSS module, was made to work in RTK mode and integrated into Mjolnir. We will start off by giving a more technical overview of network RTK and give a brief introduction to the Robot Operating System (ROS), which the rest of the sensor rig is built upon, before we go into the GNSS implementation details.

Network Real Time Kinematic Corrections

There are several commercial GNSS correction service providers, but the two major ones in Norway are *HxGN SmartNet*[12] and *CPOS*[13] and here the former was used. There also exists some free community driven base station networks, such as *RTK2GO*[14], though they can have limited coverage in some areas and are usually not as reliable as a commercially driven network.

The standard communication protocol for transmission of GNSS correctional data over the internet is referred to as Networked Transport of RTCM via Internet Protocol (NTRIP) [114]. A full overview of a Network Real-Time Kinematic (NRTK) system is best explained by an illustration as given in Figure 3.18. In a NRTK system, the moving receiver is called a *rover* and has to maintain a *NTRIP client* to communicate with the correction service. There are two primary tasks that the NTRIP client is responsible for. The first is to send the current rover position to the correction service. The second task is to listen for incoming correctional data, called RTCM messages, that are sent by the correction service.

On the other side of the communicating link, the server that the NTRIP client communicates with is called a *NTRIP caster*. The NTRIP caster receives GNSS data from a base station (or in general multiple base stations) and calculates the suitable corrections for each satellite to the rover based on its current position [114].

The Robot Operating System

A detailed overview of ROS is outside the scope of this thesis, but a brief introduction to a few of the concepts is given here to explain the later implementation details. ROS is an open-source set of software packages and different tools, which has become de facto standard for development of robotic systems [115]. One of the most important parts of ROS, is that it defines a (relatively) standardized communication interface between the users, more specifically called nodes, in the system. ROS divides the ROS nodes into *publishers* and *subscribers*. A ROS publisher can publish messages of predefined type to a named communication channel called a ROS *topic*. Meanwhile, the ROS subscribers can listen to the messages from the ROS publisher by subscribing to the given ROS topic.

[12] https://hxgnsmartnet.com/
[13] https://www.kartverket.no/til-lands/posisjon/hva-er-cpos
[14] http://rtk2go.com/

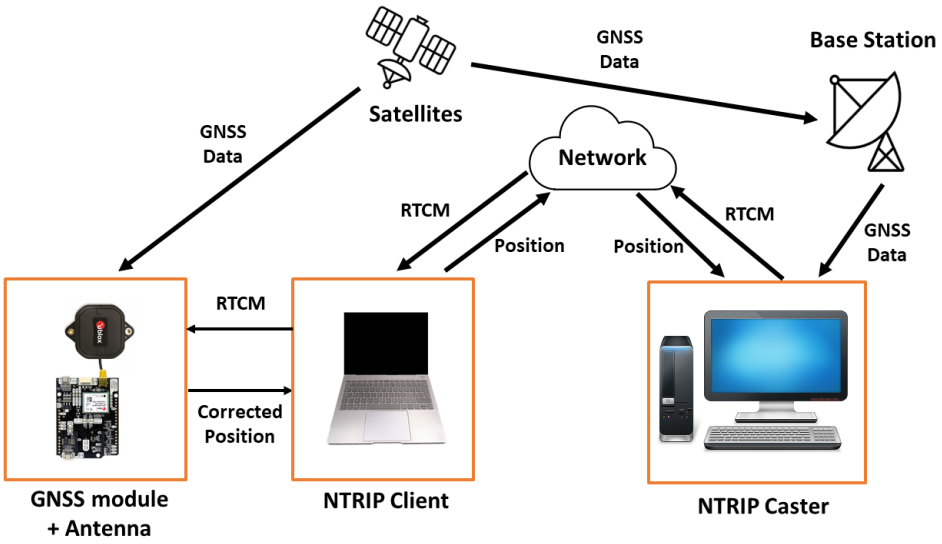


Figure 3.18: Overview of a network RTK system.

Implementation

To integrate the GNSS receiver with the rest of the *Mjolnir* sensor rig, it had to be made ROS compatible. For this purpose, an open-source ROS driver by Kumar Robotics[15] for u-blox GNSS receivers was used. An important note is that this ROS driver does not provide a NTRIP client. A challenge with this is that the simpleRTK2B board only has one USB port for communication with the ZED-F9P module and that the USB interface does not allow for simultaneous communication by multiple applications on the same USB port. Setting up an external NTRIP client that relays the correction data directly to the ZED-F9P module was therefore not an option, since this would result in collisions with the u-blox driver.

In order to solve this issue, a NTRIP client was made as a separate ROS node as shown in Figure 3.19. To acquire the position of the rover in the NTRIP client, the u-blox ROS driver formats the GNSS data from the ZED-F9P module into a ROS *NavSatFix* message and publishes it to the "/ublox/fix" ROS topic.

The NTRIP client subscribes to the same topic in order to extract the position of the rover. Since the NTRIP protocol specifies that the position of the rover must be sent to the NTRIP caster in the form of a NMEA GGA sentence[16], the NTRIP client converts the data from the NavSatFix message into a NMEA GGA sentence and sends it to the NTRIP caster over the internet as a HTTP request.

Simultaneously with updating the NTRIP caster about the rovers position, the NTRIP client listens for incoming GNSS corrections in the RTCM format from the NTRIP caster. When a RTCM message is received, the NTRIP client converts the data into a *rtcm_msgs* ROS message and sends it over a ROS topic named "/rtcm",

[15]https://github.com/KumarRobotics/ublox
[16]http://lefebure.com/articles/nmea-gga/

which the u-blox driver node subscribes to. When a RTCM message is received by the u-blox driver, the driver will forward the RTCM data to the ZED-F9P GNSS module, which leverages the data to calculate a corrected position with an internal RTK solution engine.

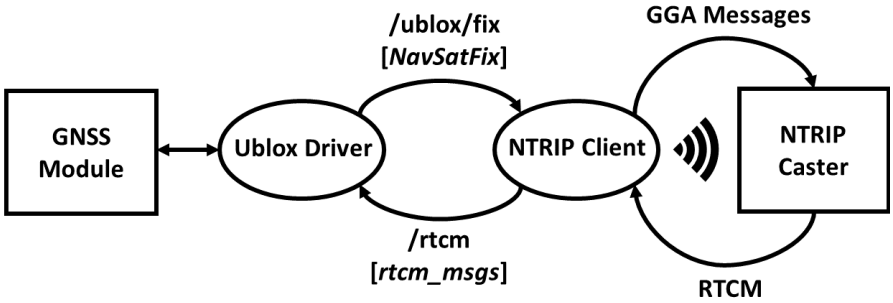


Figure 3.19: Overview of the message handling for the RTK GNSS system. A ROS node, *Ublox Driver*, communicates with the GNSS module over USB and formats the incoming GNSS data as a NavSatFix ROS messages containing the rover position and sends them to the "/ublox/fix" ROS topic. These messages are received by a second ROS node, *NTRIP Client*, which converts the position data into NMEA GGA message which is sent to the NTRIP caster over the internet. Simultaneously the NTRIP caster sends back RTCM corrections to the NTRIP client ROS node, which forwards the RTCM data as a ROS rtcm_msgs message over the "/rtcm" ROS topic. These messages are received by the *Ublox Driver* ROS node, which relays the RTCM corrections to the GNSS module.

3.9 Data Collection

Here we go through the data collection using the Mjolnir sensor rig. As we saw in Section 3.8, both the place recognition algorithm and one of the super resolution networks, require ambient images. This resulted in a few hard restrictions on the environment for where and when the data could be collected, since ambient images only result in reasonable images in good lighting conditions. To meet this requirement, the data collection environment was restricted to being outdoors during the day in clear weather conditions.

One aspect to be noted is that the data collection for the super resolution networks and the place recognition algorithm could be split into two parts. The first part was to collect data to train the super resolution networks and the place recognition visual vocabularies, and this only required the lidar images. The second part was to collect data to test the super resolution networks and the place recognition algorithm. However, in this case the test set had to include data to verify the place recognition algorithm. Therefore the place recognition test also required GNSS data for verification.

3.9.1 Training data

When collecting the lidar training images for the super resolution networks and the visual vocabularies, the focus was to gather as varied data as possible. To accommodate this, the lidar images were collected by walking around with the lidar mounted on the Mjolnir sensor rig at multiple different locations in the city of Trondheim. Note that at the time of collection, there were some regions that were partly covered in snow, but most areas were not.

3.9.2 Testing data

The testing data for lidar super resolution were collected the same day as the training data, but at different locations in Trondheim city. The weather conditions throughout the day remained relatively similar. At this point in time, the GNSS receiver needed to verify the place recognition algorithm was not fully integrated into the system yet and the place recognition test set was therefore not collected on the same day.

After the GNSS integration was finished, the place recognition test set was collected by walking with Mjolnir on a predetermined route in Trondheim, while recording lidar images at 10 Hz together with GNSS data in a single session. In total there were four place revisits over 46 minutes, spanning approximately 4 km in urban environments. At the time of collection, there was no snow and more sun than in the visual vocabulary training set described in Section 3.9.1. For the remainder of the thesis, we refer to the place recognition test set with lidar and GNSS data as *Trondheim Urban*.

3.10 Place Recognition Experiments

Here we will go through the experimental procedure for the VPR method from Section 3.2. First, in Section 3.10.1 we cover how the visual vocabularies used in the VPR method were trained. Then, in Section 3.10.2 we go through how the VPR method was tested on the collected Trondheim Urban dataset. Finally, in Section 3.10.3, we cover how the performance of the VPR method was assessed.

3.10.1 Visual Vocabulary Training

The two ORB visual vocabularies described in Section 3.2.1, which were used by the VPR algorithm, were trained on 4936 pairs of ambient and intensity images. These pairs of images were extracted from the training dataset described in Section 3.9.1. The parameters for both vocabularies were the same, using a branching factor of 10 and 5 depth levels, the L_1 score from Equation (2.2) as the score metric and TF-IDF from Equation (2.1) as the weighting function.

3.10.2 Trondheim Urban Testing

To evaluate the VPR algorithm on the Trondheim Urban dataset, the trained ambient and intensity vocabularies described in Section 3.10.1 were loaded into the two databases DB_{Amb} and DB_{Int} respectively. The selected number of ORB features to extract was set to $N_f = 500$. The maximum number of images to retrieve when querying the two databases was set to $N_s = 5$, which had to be at least $T_{min} = 60$ seconds back in time. The selected score thresholds were set to $\alpha_A = 0.05$ and $\alpha_I = 0.04$ for the ambient and intensity images respectively, while the parameter for when a new keyframe should be added was set to $\tau = 6$ seconds. In the temporal consistency check used to validate potential loop-closures, the minimum number of successive frames fulfilling the scoring criterion was set to $M = 5$.

3.10.3 Place Recognition Evaluation

To numerically evaluate the performance of the VPR algorithm, the precision defined as

$$Precision = 100\% \times \frac{TP}{TP + FP}, \tag{3.14}$$

was used, where TP is the number of correctly identified revisits and FP is the number of incorrectly identified revisits. Since we here only seek to detect a revisit and not the precise position, each detected revisit were manually evaluated to determine if it was correct or not, rather than a distance threshold. To qualitatively evaluate the VPR algorithm, the detected revisits were inspected to see where the algorithm is able to detect a revisit, where it potentially failed to detect a revisit and where it erroneously reported a detected revisit.

It is also interesting to identify in which cases the method is able to leverage the combination of the ambient and intensity images, to possibly be more robust to perceptual aliasing occurring in only one of the images. However, since the VPR method is based mainly on a voting-based scheme between the ambient and intensity images to detect or reject a potential revisit, it is not trivial to evaluate how the method would work with only one of the images, since this would essentially invalidate the basic premise of the method. Here it was therefore evaluated in which instances either an intensity or ambient image query resulted in a high similarity score to an image from a different location in their respective database. One issue with this method, is that the similarity score is dependent on the specific visual vocabulary that is used when comparing the images, so the score itself is not a very meaningful number. It was therefore decided to manually evaluate when the similarity score in a given instance was relatively high compared to what was observed in the rest of the Trondheim Urban dataset.

3.11 Lidar Super Resolution Experiments

Here we cover the experimental procedure for how the range, intensity and ambient super resolution networks described in Sections 3.5 and 3.6, were trained,

tested and evaluated. The section is organized in the following way. First in Section 3.11.1 we go through how the networks were trained and show e.g. which specific hyperparameters that were used. In Section 3.11.2 we cover the testing procedure for the lidar super resolution networks. After this, we go through how the performance of the super resolution performance was evaluated, which is split into two parts. Section 3.11.3 presents how the range super resolution network was evaluated. Then in Section 3.11.4, we present the evaluation methods for the ambient and intensity super resolution networks together. The reason for this separation, is that the evaluation methods for the range super resolution network are relatively different from those that are used to evaluate the ambient and intensity super resolution networks.

3.11.1 Training Procedure

All the super resolution networks (range, intensity and ambient) were trained to upscale from 32 to 128 channels and from 64 to 128 channels, using the TensorFlow machine learning library [116]. Similar to the lidar super resolution paper by Shan et al. [79], the networks were trained using the mean of absolute differences between the predicted pixel values and the true pixel values as the loss function. All three networks were trained using the Adam optimizer [102] and initialized with a learning rate of 10^{-4}. For the range super resolution network, the learning rate was reduced with an exponential decay rate of 10^{-5}. The ambient and intensity networks were also trained with the Adam optimizer with an initial learning rate of 10^{-4}, but unlike the hyperparameters used by Shan et al. [79], the learning rate was halved after every 2×10^5 iterations. Due to constraints on the available GPU memory, the batch size for all three networks had to be set to 2. The dropout rate [103] was set to 0.25 for all three networks. The data augmentation applied to the training data consisted of random horizontal flipping and random circular shifting of the images, to mimic changes in viewpoint.

For the range super resolution network, the final model was similiar to Shan et al. [79] selected as the one that had the minimum loss on the validation set. However, since the ambient and intensity images are more similar to normal camera images, the final ambient and intensity models were selected as the ones that maximized the PSNR metric from Equation (2.7) on the validation set.

For the range noise filtering described in Section 3.3.1, the number of forward passes during inference was set to 16, while the threshold parameter from Equation (3.4) was based on tuning set to $\alpha = 0.005$.

3.11.2 Testing Procedure

To test the super resolution networks, the range, intensity and ambient images from the collected lidar super resolution test set described in Section 3.9.2, were used. These images were downscaled from an original resolution of 1024×128 to a resolution of 1024×64 and 1024×32, by extracting every second and fourth row

respectively, to simulate two individual lidars with 64 and 32 channels. The downscaled images were then used as input to the trained super resolution networks, which resulted in predicted images with the original 1024×128 resolution.

3.11.3 Range Super Resolution Evaluation

To quantitatively evaluate the range super resolution network, the original and predicted range images described in Section 3.11.2 were scaled so that the pixel values had meters as the unit, similar to what we saw in Equation (3.5). The pixels with a value above 50 meters, corresponding to the maximum specified range of the OS0-128 lidar, were set to 0 meters in both the predicted and original range images, since these were here regarded as invalid values. Similarly, all pixel values within 1 meter from the lidar were set to 0 meter, since these measurements mainly corresponded to noisy reflections from the platform the lidar was mounted on.

Evaluation Metrics

After the former pre-processing steps, the sum of the absolute difference between the range values $\hat{x}_i$ in the predicted range images and the range values x_i in original range images, were computed. This is mathematically given as

$$e_i = |x_i - \hat{x}_i|, \quad i \in \{1, \ldots, N\} \tag{3.15}$$

where e_i is the absolute range error for prediction i and N is the total number of predictions in the test set, but only the range errors where the predicted range image had valid prediction in the sense that the pixel value was non-zero, were extracted. Since there could be significant outliers in the range predictions, it might be that the mean range error is not a well-suited evaluation metric. For this reason, both the Mean Average Error (MAE) calculated as

$$MAE = \frac{1}{N} \sum_{i=1}^{N} e_i \tag{3.16}$$

and the Median Average Deviation (MAD) corresponding to median of all the absolute range errors are reported, since the MAD is known to be robust to outliers. To evaluate which of these two metrics is best suited to represent the range error of the super resolution network, histograms of the absolute range error for every pixel in the range image test set from 32 to 128 channels and from 64 to 128 channels were created, so that the error distribution could be examined.

Single Location Evaluation

To evaluate the range super resolution network from a qualitative standpoint, the prediction from 32 to 128 channels and 64 to 128 channels using the range super resolution network at a location in the lidar super resolution test set described in

Section 3.9.2, was analysed. Although the predictions from the range super resolution network are range images, we mainly present the results as point clouds using the method from Section 3.5.2 for visualization purposes. At this location, the effects of the uncertainty noise filtering described in Section 3.3.1 was examined and the overall quality of the filtered point clouds was assessed.

Challenging Location Evaluation

As described in Section 3.3.1, the noise filtering proposed in the lidar super resolution paper by Shan et al. [79] seeks to remove uncertain points from the predicted point cloud, e.g. points that are located close to edge boundaries. One would expect that one of the more challenging scenarios for the range super resolution network would be to make correct predictions for edges in the range image where there is only a short range gap. In particular, since the range images were only upscaled in the vertical direction, we would expect that it is especially challenging for the range super resolution to make correct predictions on the boundary of horizontal edges in the range image. It was therefore evaluated how the range super resolution network performed for an instance from the test set described in Section 3.9.2 containing a horizontal edge with a short range gap.

Person Detection Evaluation

To evaluate if the range super resolution network could potentially be used in e.g. object or person detection, an instance from the lidar super resolution test set described in Section 3.9.2 containing a walking person was examined. The downsampled point cloud of the person from 128 channels to 32 and 64 channels were compared with the upsampled point clouds using super resolution.

3.11.4 Ambient and Intensity Super Resolution Evaluation

To evaluate the performance of the intensity and ambient super resolution networks quantitatively, the average PSNR and SSIM on the entire lidar super resolution test set described in Section 3.9.2 were computed. Here the PSNR and SSIM values were calculated using the functions *psnr* and *ssim* in MATLAB. The PSNR and SSIM results of the super resolution networks were compared with upsampled images using bicubic interpolation, which were created using the function *imresize* in MATLAB. A qualitative evaluation of the super-resolved ambient and intensity images was also performed and compared with the result of using bicubic interpolation.

Intensity Colored Point Cloud

Since there is a one-to-one correspondence between the pixels in the range, intensity and ambient images, it was tested to generate a 3D point cloud based on

a predicted range image using the method described in Section 3.5.2. The one-to-one correspondence between the range and intensity image was then used to colorize each point in the point cloud with the values from the predicted intensity image from the same location. Finally, the overall quality of the intensity colored point cloud was examined, to evaluate if it would be possible to combine the information from the super-resolved range and intensity image by this method.

3.12 High Resolution Image Generation Experiments

Here we go through how the lidar image generation pipeline described in Section 3.7, was tested. The pipeline was tested on two different datasets from different environments; outdoor and indoor, which is covered in Section 3.12.1 and Section 3.12.2 respectively.

3.12.1 Outdoor Images - Newer College

The first test was performed on the public dataset *Newer College* by Zhang et al. [113], since it provides data from an OS0-128 lidar. The dataset is collected outdoor at walking speed and it also provides a ground truth trajectory with centimeter accuracy. In the test, the following operation was performed on multiple different locations in the dataset: First a point cloud C_i at a given time instance t_i was selected as the point cloud reference. Then the 3 preceding and succeeding point clouds to C_i were extracted together with C_i so that the total point cloud set $\mathcal{C}$ consisted of 7 point clouds in total, which were used as input to image generation pipeline. As an initial alignment for the ICP refinement described in Section 3.7.2, the ground truth trajectory provided by the dataset was used. The images were selected to have a resolution 1024×256, where the vertical resolution of 256 is twice the number of channels available by the lidar, while the horizontal resolution of 1024 is the same as what the lidar images had originally.

3.12.2 Indoor Images - Entrance Hall

The second test was performed on a self-collected dataset from an entrance hall, where the Mjolnir sensor rig with the OS0-128 lidar, was slowly rotated in the vertical direction, while the translational movement was kept to a minimum. Similar to the test on the Newer College dataset, a point cloud C_i at a given time instance t_i was selected as the point cloud reference. However, since the relative motion between each cloud is significantly smaller than in the Newer College dataset, 10 preceding and succeeding point clouds to C_i were extracted, so that the point cloud set $\mathcal{C}$ used as input to the pipeline consisted of 21 point clouds in total. Data from an IMU was also recorded to provide an initial alignment for ICP, but the data remained unused since the point cloud registration converged sufficiently by choosing the initial transformations equal. The lidar images were generated at two different resolutions, 1024×256 and 1024×512, which corresponds to an

upscaling in the number of channels by 2× and 4× respectively, compared to the original lidar images.

Chapter 4

Results and Discussion

This chapter covers the results from the experiments in Chapter 3. First, Section 4.1 goes through the results of the VPR algorithm and the related Trondheim Urban dataset. Then, Section 4.2 covers the results of the lidar super resolution networks. Finally, Section 4.3 covers the results of generating high resolution lidar images based on multiple point clouds.

4.1 Visual Place Recognition

Here the results of testing the VPR algorithm from Section 3.2 is covered. The section is organized as follows: Section 4.1.1 presents the collected Trondheim Urban dataset described in Section 3.9.2, which was used to test the VPR algorithm. In Section 4.1.2, the results from testing the VPR algorithm described in Section 3.10 are presented, together with a discussion of the results.

4.1.1 Trondheim Urban Dataset

An overview of the route based on the recorded GNSS data from the collected Trondheim urban dataset is shown in Figure 4.1. In total, the dataset contains four place revisits. Each of the four place revisits are highlighted in Figure 4.2 with arrows indicating the direction of the revisit. The indicated numbering of the four revisits is referred to as location 1, 2, 3 and 4 in the following discussion. Note that at location 2, the revisit is in the opposite direction, which can be a challenge for many place recognition algorithms [117].

An important observation for the dataset, is that it turned out to be hard to maintain the GNSS module in RTK mode, which resulted in some "jumps" in the GNSS measurements as shown in Figure 4.3, when the GNSS module switched between standard GNSS positioning and RTK positioning. This is likely due to high signal blockage from surrounding buildings, since the dataset is collected in an urban environment, as well as a consequence of the angled placement of the GNSS antenna on the sensor rig as we saw in Figure 3.17.

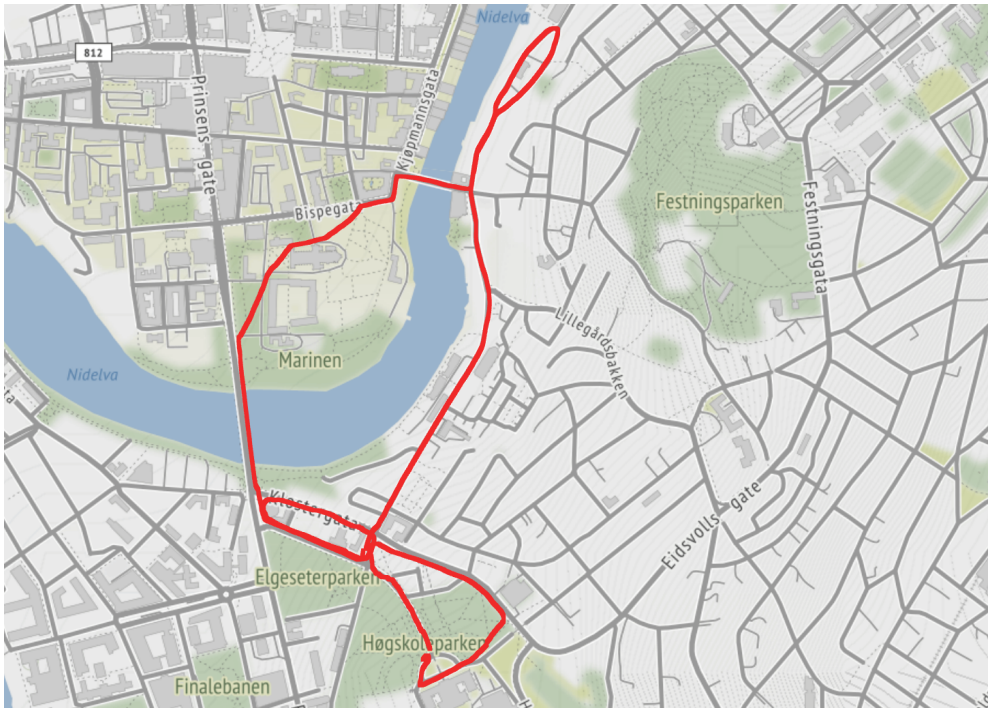


Figure 4.1: Map showing the route of the Trondheim urban dataset based on the recorded GNSS data. The dataset is recorded over 46 minutes, spanning approximately 4 km. Map tiles by © Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under ODbL.

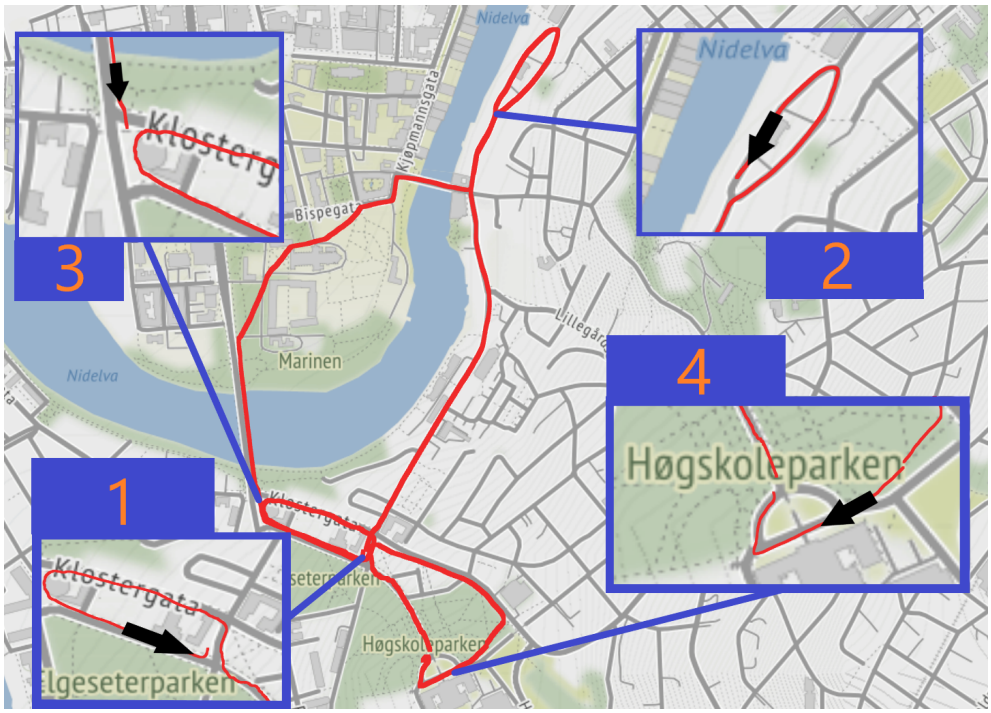


Figure 4.2: The four place revisits in the Trondheim Urban dataset, where the arrows indicate the direction of the revisit, while the numbering corresponds to the chronological order of the revisits. Map tiles by © Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under ODbL.

Figure 4.3: An example from the Trondheim Urban dataset where the GNSS data "jumps" when the GNSS receiver converts from standard GNSS positioning (red) to RTK positioning (green). Map tiles by © Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under ODbL.

4.1.2 Place Recognition Results

Here we cover the results of the VPR algorithm using the evaluation methods described in Section 3.10.3. A summary of the numerical results of the VPR algorithm tested on Trondheim Urban is shown in Table 4.1. As we can see from the table, the VPR algorithm correctly identified 18 place revisits, while 2 of them were incorrect, which resulted in a precision of 90%. Note that even though there are only 4 revisits in the Trondheim urban dataset, there can be multiple detections of the same revisit, since they last for an extended period. However, the VPR algorithm was able to detect all four revisits at least once.

Table 4.1: The number of true and false positives for the revisits detected by the VPR algorithm and the resulting precision on the Trondheim Urban dataset.

True Positives	18
False Positives	2
Precision	90%

Correct Revisit Detection

An example of a correct detection of the reverse revisit from location 2 in the Trondheim urban dataset is shown in Figure 4.4. Here we see that although there is a significant horizontal shift between the images, since they are captured when approaching the same location in opposite directions, the VPR algorithm is able to detect the revisit. This result is likely due to the 360° field of view of the lidar, combined with the fact that BoVW is based on evaluating the frequency of the visual words in an image and not their position. We would therefore expect that the algorithm is largely invariant to these shifts.

Incorrect Revisit Detection

An example where the place recognition algorithm erroneously detects a revisit is shown in Figure 4.5. Note that the second incorrect revisit detection was located only a few meters from the one that is shown here, so the following discussion is relevant for both of them. By observing the images, it seems like the VPR algorithm detects it as a revisit since there are relatively few structures in the area while the sun stands out as a prominent point in both the ambient and intensity images. Also, if we look at a map of the actual location of the lidar and where the VPR algorithm detected a revisit as shown in Figure 4.6, we see that both locations are actually in the same region of the map. When combined with the fact that there are few buildings in the region, the lidar images do likely have a partial overlap in the perceived horizon. This could therefore be a contributing factor to why it is detected as a revisit.

(a) Ambient query image.

(b) Intensity query image.

(c) Retrieved location from the ambient database.

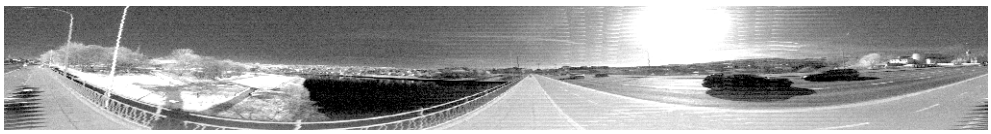
(d) Retrieved location from the intensity database.

Figure 4.4: An example where the place recognition algorithm is able to correctly identify the revisit of location 2 from the Trondheim Urban dataset. In (a) and (b) we see the ambient and intensity query images, which the place recognition algorithm is able to correctly couple with the ambient (c) and intensity (d) images from their respective database. We can see that even though the revisit is in the reverse direction, the algorithm is able to detect it, which is likely due to the 360° field of view of the lidar. Note that the dark regions at the left and right side of the intensity images is because of the power cable to the lidar and not the location itself.

Perceptual Aliasing Robustness

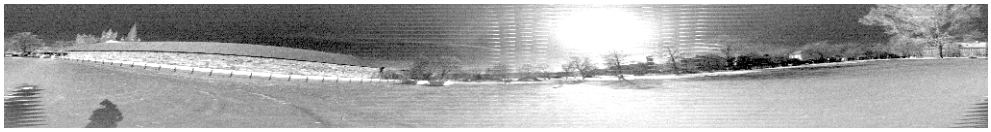
In Figure 4.7 we see an interesting instance from the Trondheim Urban dataset, where an ambient image query resulted in a similarity score to an image in the database that was more than three times higher than the ambient score threshold, even though these images are from two completely different locations. Based on the images, it seems like they receive a high similarity score since both images contain a building that is partially blocking the sun. However, since the intensity image query and the image from the intensity database at the same locations shown in Figures 4.7b and 4.7d appears to be clearly different, this was not retrieved as one of the highest scoring images after performing a query to the database. The VPR algorithm was therefore able to filter away the spurious ambient match, since it was not part of the intersection between the sets of retrieved ambient and intensity images.

Likewise in Figure 4.8 we see the opposite instance, where the intensity query

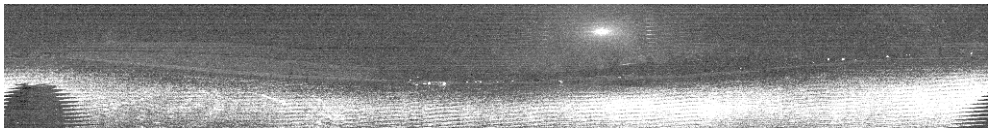


(a) Ambient query image.

(b) Intensity query image.



(c) Retrieved location from the ambient database.



(d) Retrieved location from the intensity database.

Figure 4.5: An example where the place recognition algorithm erroneously detects a revisit. In (a) and (b) we see the ambient and intensity query images, which the place recognition algorithm incorrectly associates with the ambient image (c) and the intensity image (d) from the databases from another location. Based on the images, it seems like the algorithm has troubles since there are relatively few structures at both locations, while the sun stands out as a distinctive point in both the ambient and the intensity images.

resulted in a similarity score with an image from the database that was more than three times as high as the intensity score threshold, despite that these images are from entirely different locations. Based on observation of the images, it is likely that these images had a high similarity score, since they both contain relatively open areas. Meanwhile, the ambient images shown in Figures 4.8a and 4.8c from the same locations are distinctively different, which is likely why these were not regarded as a match when performing the database query, so that the VPR algorithm could filter it away.

Based on these observations, it seems like the VPR algorithm is able to leverage a combination of the ambient and intensity images for increased robustness to spurious matches in a single lidar image pair. Still, given that the quality of the ambient image is highly dependent on good lighting conditions and can degrade significantly even as a result of cloudy conditions, this does obviously limit the applicability of the algorithm. In a real situation, it would be unreasonable to assume that we always operate during daytime in clear weather conditions. Yet, these results indicate as a proof of concept that the ambient image can be utilized

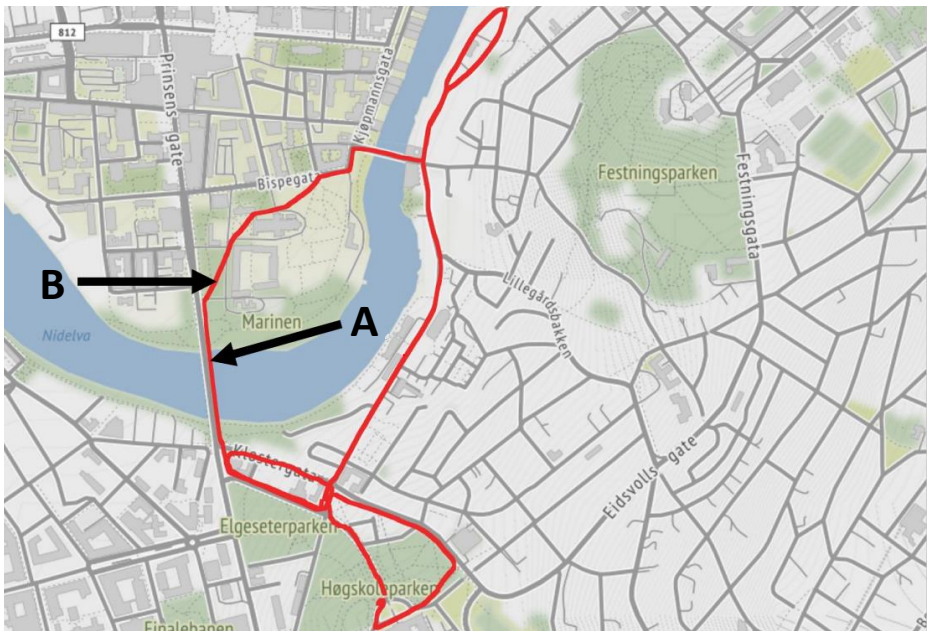
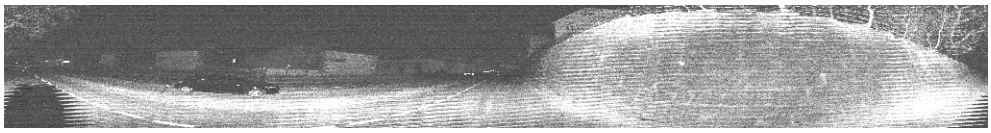


Figure 4.6: In the map we see the real location of the lidar (A) and the location the VPR algorithm erroneously detects a revisit for (B). The distance between the two locations is approximately 180 meters. We can observe that although it is not a revisit in the sense that we have returned to the same location, the detected revisit is located in the same region of the map. When combined with the fact that the region has relatively few buildings resulting in partially overlapping horizons, this could be a part of the explanation for why the algorithm detects a revisit. Map tiles by © Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under ODbL.

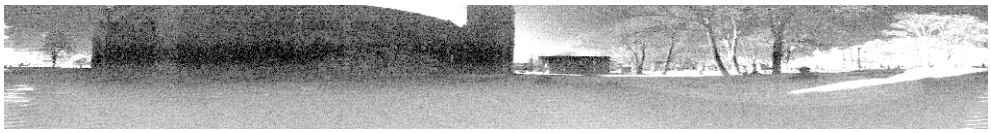
in a complementary manner to the intensity image, though a full scale implementation would have to factor in the uncertainty in the ambient image based on the conditions the algorithm is operating under.

A drawback of this algorithm, is that as a consequence of using voting to determine a revisit, it can potentially be problematic to detect multiple revisits to the same location. This is due the fact that the algorithm could end up retrieving images from different time instances from the same location from the two databases. This would result in a non-overlap in the intersection of the retrieved image sets and therefore would not be regarded as a revisit. One potential solution to this problem is to group together earlier detected revisits, so that they can be associated to the same location. Another way to mitigate this effect, could be to make an assumption for the maximum time until a revisit. This would make it possible til filter away earlier revisits, which would reduce the probability of this event occurring.

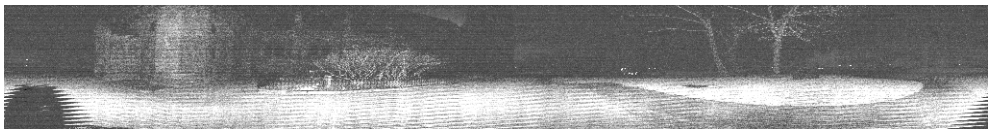
(a) Ambient query image.



(b) Intensity query image.



(c) Ambient database image.



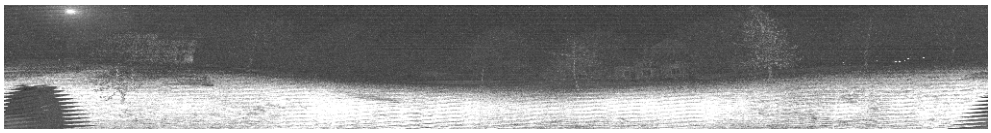
(d) Intensity database image.

Figure 4.7: An instance from Trondheim Urban where an ambient query image (a) appears visually similar to an image from the ambient database (c), which resulted in a high similarity score after executing a database query. These images are in reality from two separate locations, which is considerably easier to see in the intensity query image (b) and the intensity database image (d) from the same locations as (a) and (c) respectively. As a result, the image (d) was not among the highest scoring intensity images after executing a database query, so the high scoring match between (a) and (c) was filtered out by the intersection stage in the VPR pipeline.

(a) Ambient query image.

(b) Intensity query image.

(c) Ambient database image.



(d) Intensity database image.

Figure 4.8: An instance from Trondheim Urban where an intensity query image (b) appears visually similar to an image from from the intensity database (d), which resulted in a high similarity score after executing a database query. These images are in reality from two separate locations, which is considerably easier to see in the ambient query image (a) and the ambient database image (c) from the same locations as (b) and (d) respectively. As a result, the image (c) was not among the highest scoring ambient images after executing a database query, so the high scoring match between (b) and (d) was filtered out by the intersection stage in the VPR pipeline.

4.2 Lidar Super Resolution

Here we cover the results of the lidar super resolution network experiments described in Section 3.11. The section is organized as follows: First in Section 4.2.1 we go through the results of the range super resolution network based on the evaluation methods described in Section 3.11.3. Then in Section 4.2.2 we cover the results of the ambient and intensity super resolution networks based on the evaluation methods in Section 3.11.4.

4.2.1 Range Super Resolution Results

An overview of the mean and median range prediction error calculated using the method from Section 3.11.3 on the lidar image test set described in Section 3.9.2, are shown in Table 4.2. Histograms of the absolute range error calculated on the same test set, together with the mean and median, are given in Figure 4.9 when upsampling from 32 to 128 channels and in Figure 4.10 from 64 to 128 channels.

Table 4.2: Mean and median range prediction error, as well as the interquartile range (IQR), for the range image super resolution network from 32 to 128 channels and from 64 to 128 channels.

Upscale	Mean Error [m]	Median Error [m]	IQR [m]
$32 \rightarrow 128$	4.2×10^{-1}	8.4×10^{-2}	9.2×10^{-2}
$64 \rightarrow 128$	2.1×10^{-1}	2.4×10^{-2}	6.0×10^{-2}

From Table 4.2 we see that the mean range error prediction error from 32 to 128 channels is more than 40 cm, which at first glance seems to be very high. However, by looking at the histogram of the error distribution in Figure 4.9, we can observe that the mean is heavily skewed due to outliers. Based on these observations it seems like the median value of 8.4 cm is a more adequate representation of the range error, though a median error of almost 10 cm does still seem relatively high, given that the specified precision of the OS-128 lidar is $\pm[1.5 - 5]$ cm (depending on the range to the object) [118]. We do nonetheless have to expect that there will be higher degree of uncertainty in the range values when predicting the range measurements, compared to using the real sensor data.

From Table 4.2 we see that the range prediction error decreases significantly when upscaling from only 64 to 128 channels instead of 32 to 128 channels. Likewise, by comparing the error histograms in Figure 4.9 and Figure 4.10, we can observe that the proportion of outliers decreases notably when predicting from 64 to 128 channels instead of 32 to 128 channels. Although it was expected that the error would become smaller in this case, it is reassuring that it evidently does decrease.

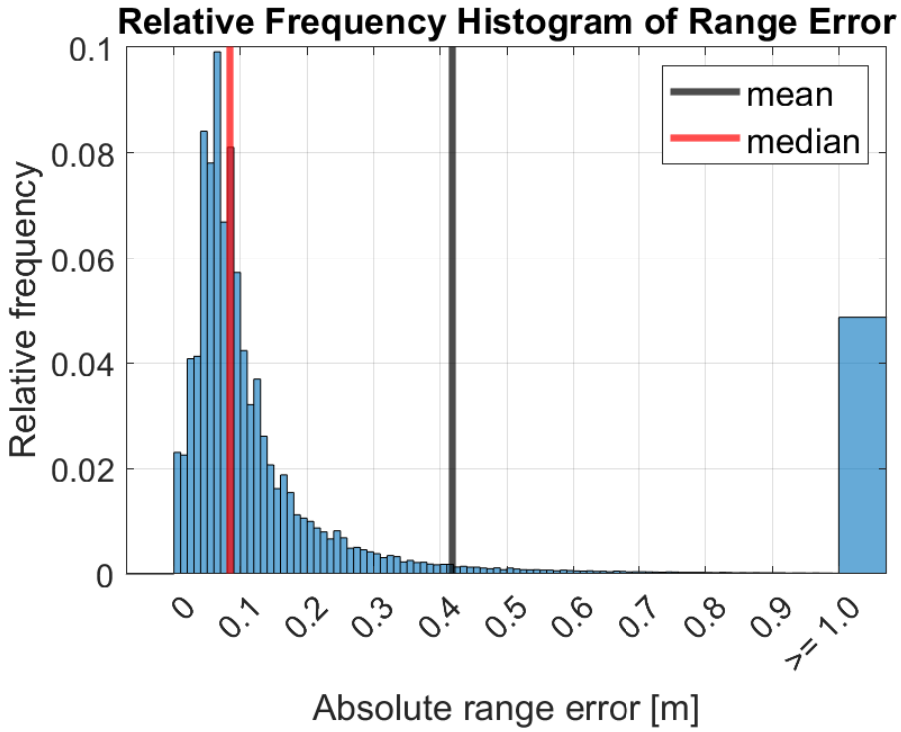


Figure 4.9: Histogram showing the relative distribution of the absolute range error for the super resolution network when upsampling from 32 to 128 channels. Since the range error can be as high as 50 meters, corresponding to the max range of the lidar, all range errors above 1 meter are grouped into a single bin. From the histogram we can observe that the mean is heavily skewed due to outliers, while the median appears to be a better representation of the distribution.

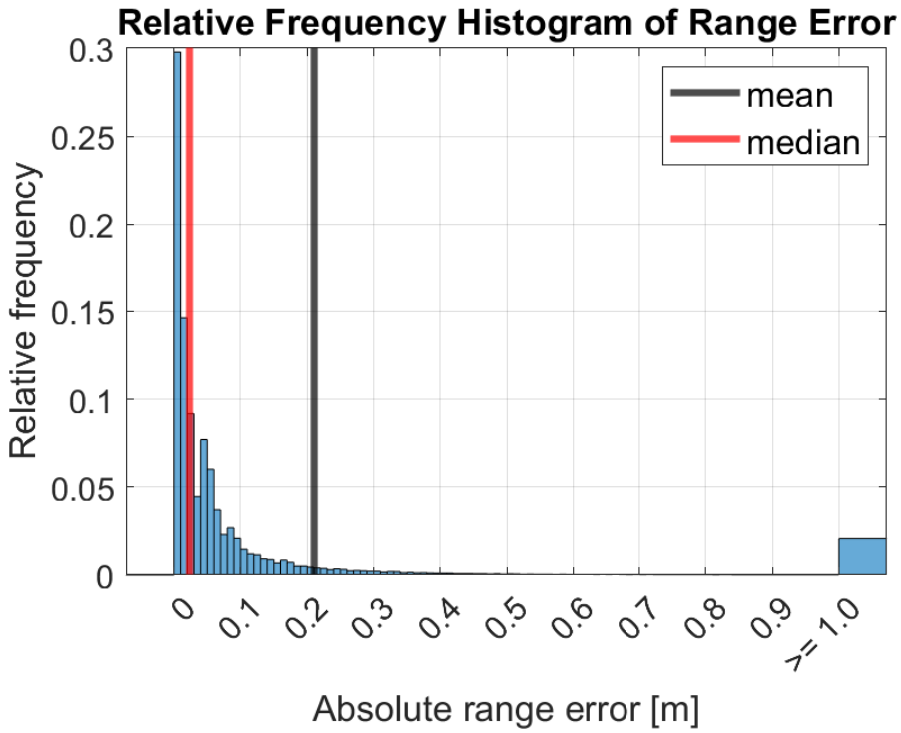
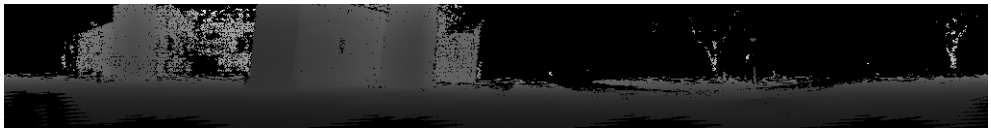


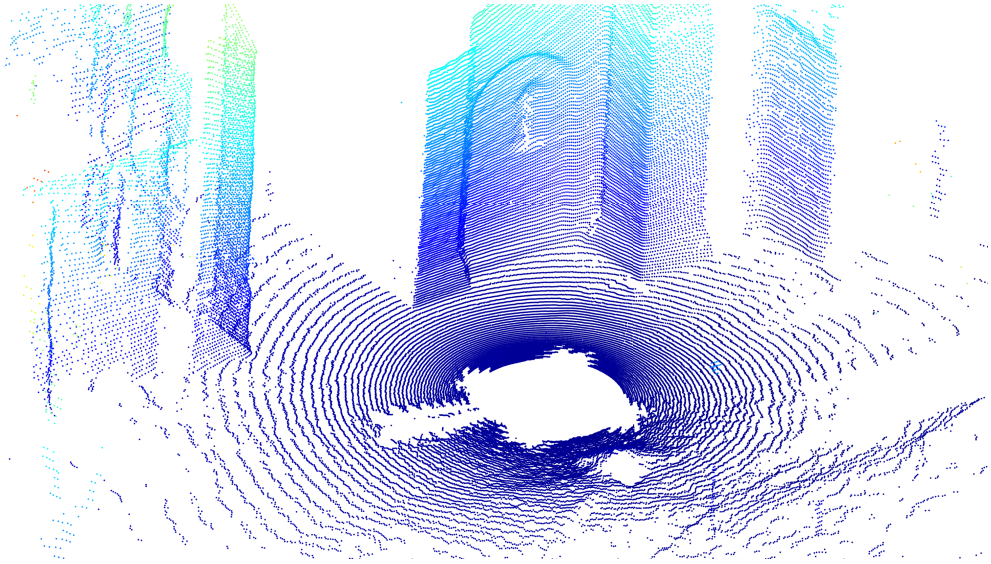
Figure 4.10: Histogram showing the relative distribution of the absolute range error for the super resolution network when upsampling from 64 to 128 channels. All range errors above 1 meter are grouped into a single bin. From the plot we see that that most range errors are close to the median, while the outliers at $>= 1$ meter causes the mean to become skewed.

Single Location Evaluation

Here the range image in Figure 4.11a is used to qualitatively evaluate the range super resolution network as described in Section 3.11.3. However, since it is easier from a visual standpoint to interpret the range image as a point cloud as in Figure 4.11b, which is generated by the method explained in Section 3.5.2, we will for the remainder of the discussion visualize the range image as a point cloud. In Figure 4.12 we can see the point cloud from Figure 4.11b downsampled to 64 channels and 32 channels, which are used as input to the range super resolution network.

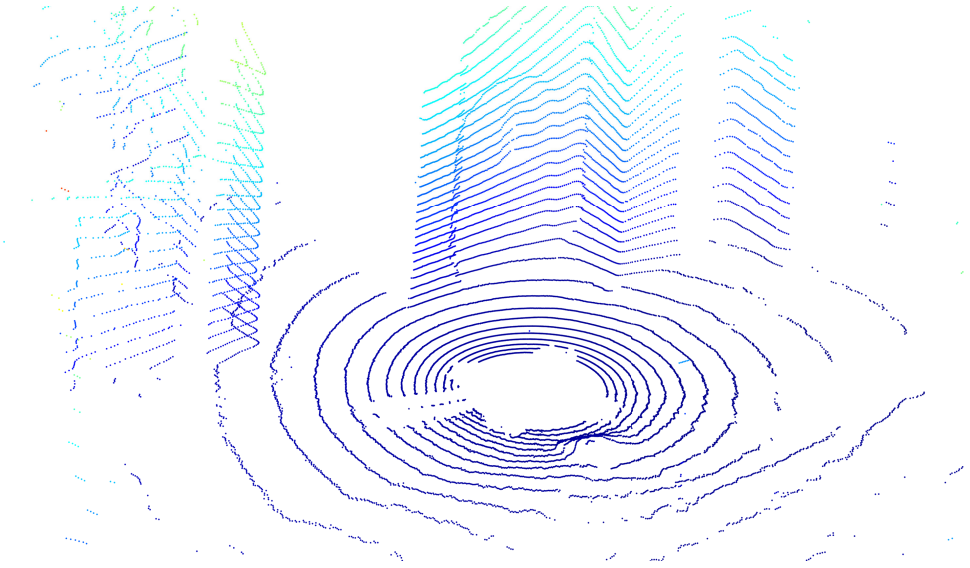


(a) Range image

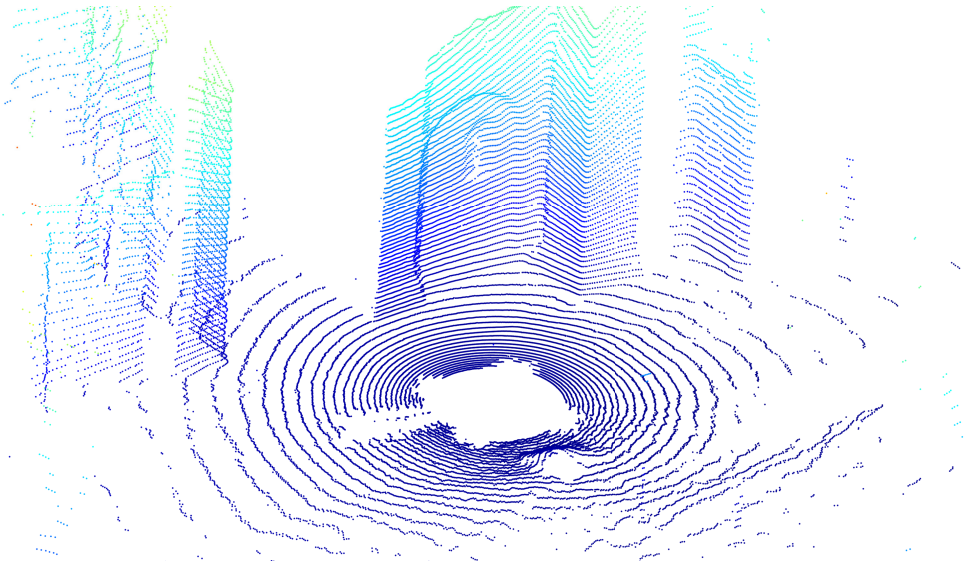


(b) Point cloud generated based on the range image

Figure 4.11: The range image (a) that is used as the ground truth for the test location and the point cloud (b) we get by projecting range image to 3D space. Since it is significantly easier to interpret the point cloud, we use that in the following discussion for visualization purposes but note that the range super resolution network actually outputs a range image and not a point cloud.



(a) Original point cloud downsampled to 32 channels

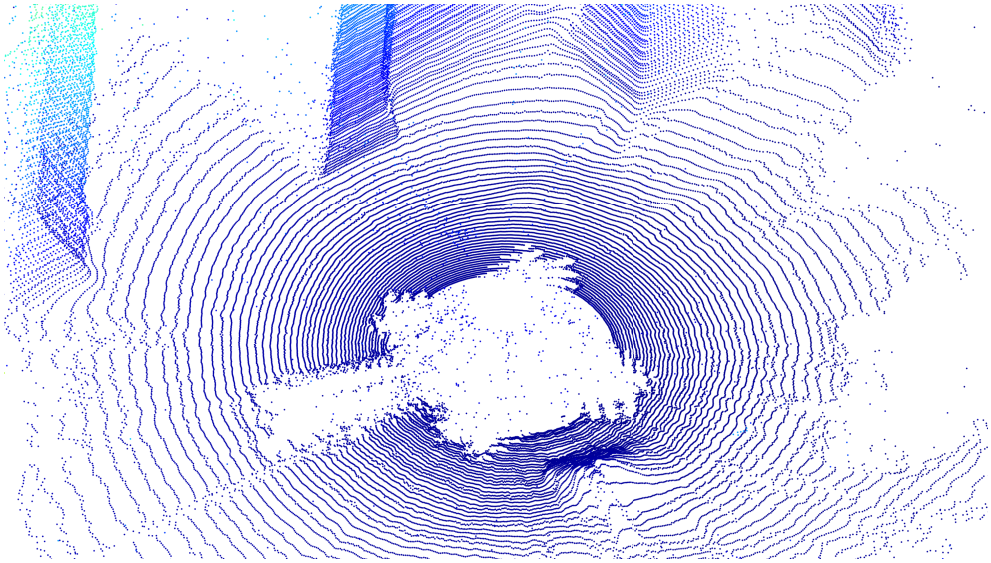


(b) Original point cloud downsampled to 64 channels

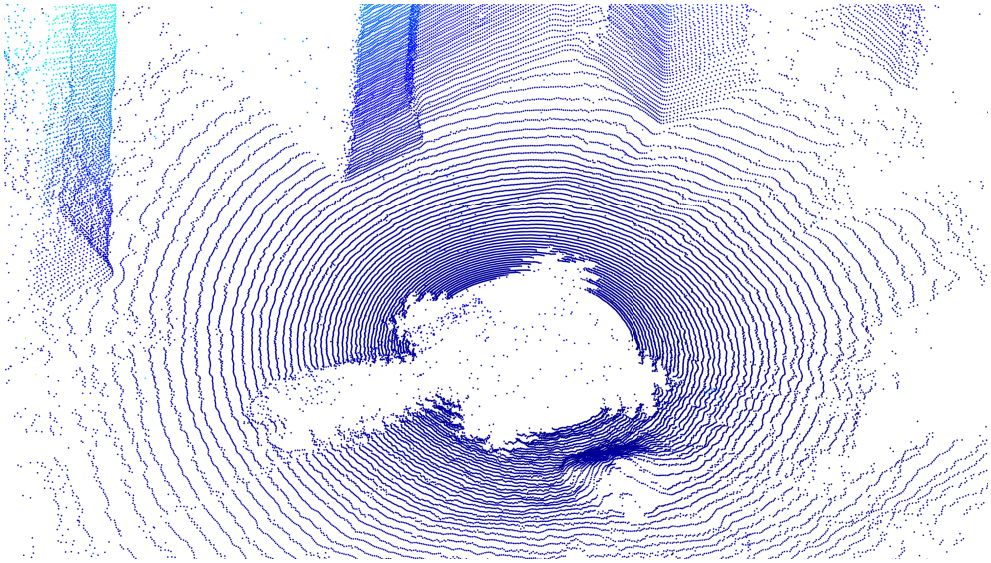
Figure 4.12: The result of downsampling the 128 channels pointcloud from Figure 4.11b to 32 channels (a) and 64 channels (b) that are used by the range super resolution network to predict a point cloud with 128 channels.

In Figure 4.13 we can see the predicted point clouds from the range super resolution network based on the input shown in Figure 4.12 for both 32 to 128 channels and 64 to 128 channels, prior to applying the uncertainty noise filtering described in Section 3.3.1. We can clearly see that both predicted point clouds are dominated by "beams" of noise, which would likely render the point clouds impractical to use for most purposes. By close inspection of the region around the center of the point clouds, we can see that there is a lower proportion of outliers in the predicted point cloud from 64 to 128 channels point cloud in Figure 4.13b compared to the predicted point cloud from 32 to 128 channels in Figure 4.13a. However, since both clouds have a significant number of outliers, this difference becomes hardly noticeable.

In Figure 4.14 we can see the same predicted point clouds as in Figure 4.13 after applying the uncertainty noise filtering described in Section 3.3.1. We can observe that there is a significant reduction in noise in both the predicted point cloud from 32 to 128 channels and from 64 to 128 channels. Note that the mostly empty "tail-like" region is mainly a result of the power cable to the lidar blocking parts of the lidar's field of view. However, an interesting detail is that after the uncertainty filtering, there have been removed more points along this region in the prediction from 64 to 128 channels shown in Figure 4.14b than in the predicted point cloud from 32 to 128 channels shown in Figure 4.14a. Intuitively we would expect that the network would be less uncertain about these points in the predicted point cloud from 64 to 128 channels than from 32 to 128 channels and hence fewer points would be removed. However, artificial neural networks can sometimes be unpredictable, so it is not easy to say why this occurs here. It might just be a random result from this input to the super resolution network.

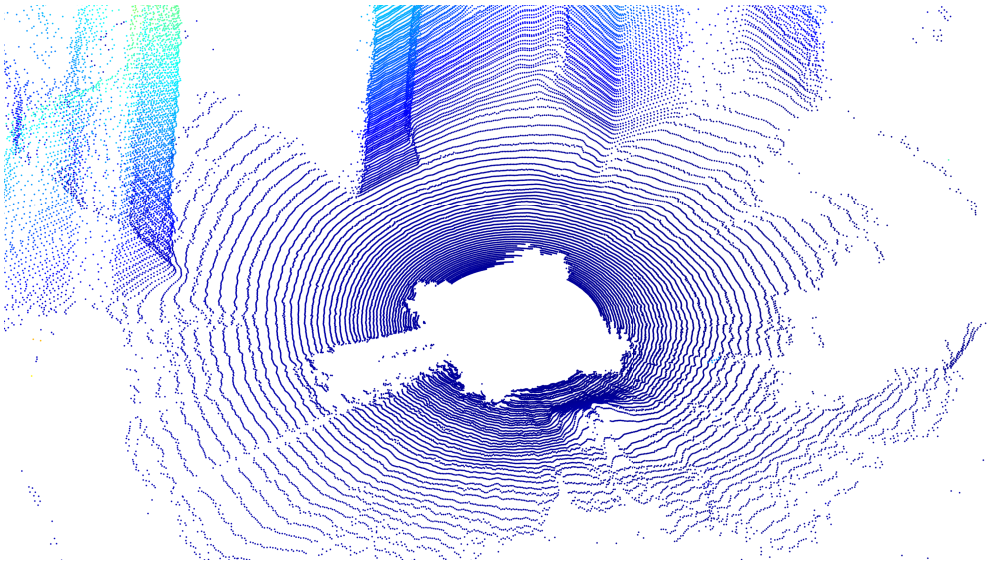


(a) Predicted point cloud from 32 to 128 channels prior to noise filtering.

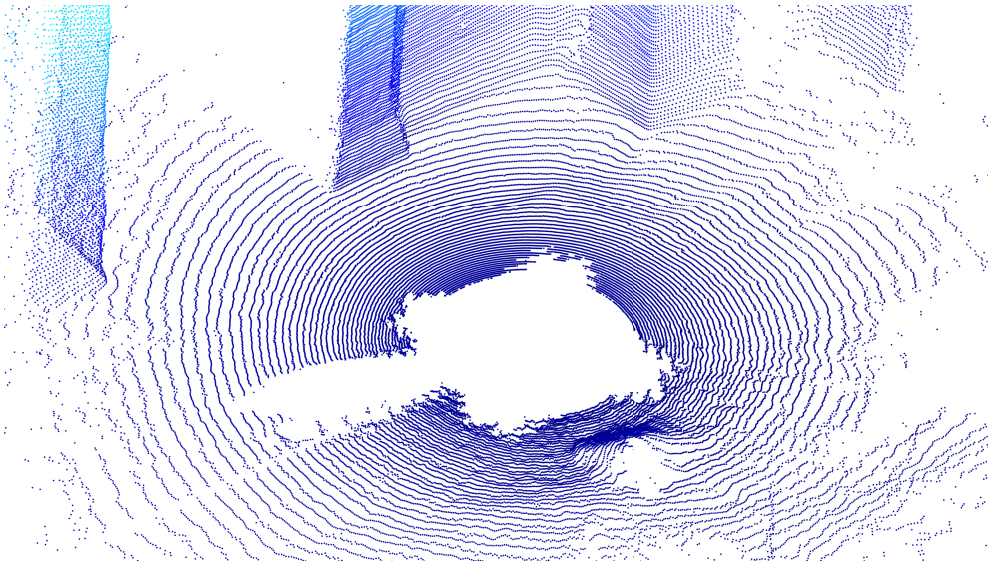


(b) Predicted point cloud from 64 to 128 channels prior to noise filtering.

Figure 4.13: The predicted point cloud from the super resolution network from 32 to 128 channels (a) and 64 to 128 channels (b) prior to applying the uncertainty noise filtering. We can see that there is a significant amount of outliers in both point clouds compared to Figure 4.11b, which is especially noticeable in the center of the point clouds.



(a) Predicted point cloud from 32 to 128 channels after noise filtering.

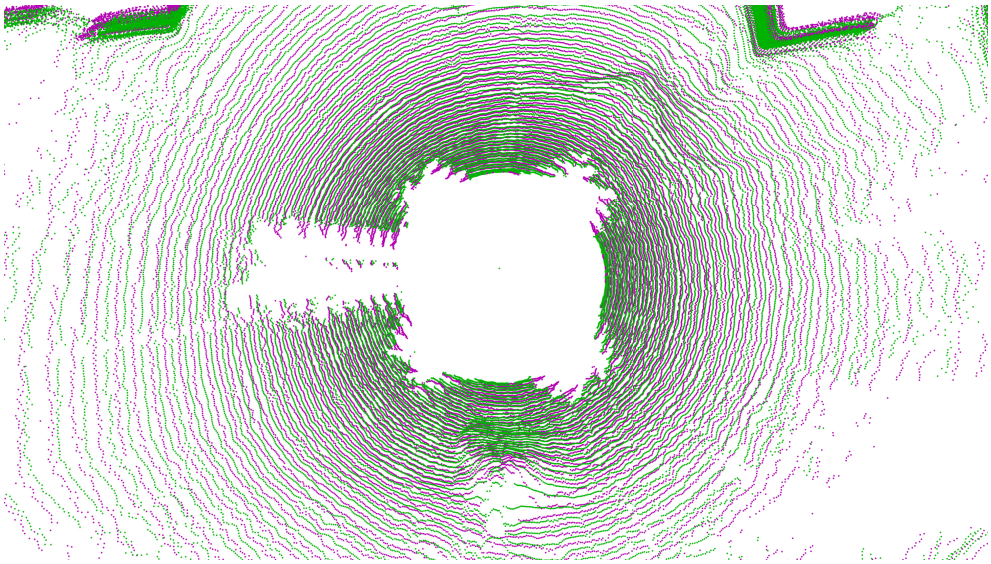


(b) Predicted point cloud from 64 to 128 channels after noise filtering.

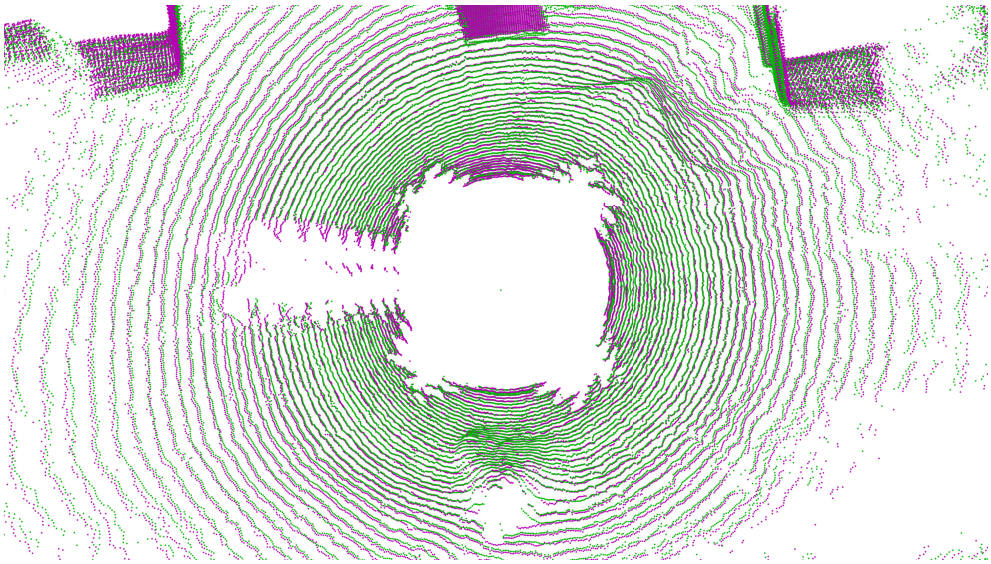
Figure 4.14: The predicted point cloud from the super resolution network from 32 to 128 channels (a) and 64 to 128 channels (b) after applying the uncertainty noise filtering. We can see that there has been a substantial reduction in noise for both point clouds compared to Figure 4.13. An unexpected result is that the empty "tail-like" region caused by the power able extends further in the predicted point cloud from 64 to 128 channels in Figure 4.14b, compared to the predicted cloud from 32 to 128 channels in Figure 4.14a.

Figure 4.15 shows the same predicted point clouds as in Figure 4.14 from a bird's-eye view, together with the original 128 channel point cloud from Figure 4.11b. We can observe that the predicted point clouds appear to match the original point cloud relatively well. As expected, we see that the 64 to 128 channels prediction shown in Figure 4.15b matches the original cloud better than the prediction from 32 to 128 channels shown in Figure 4.15a. Another important observation is that the mismatch between the predicted and original point cloud apparently increases with range. A likely explanation for this, is that the distance between the "rings" in the point cloud increases with the range from the lidar. As a consequence there will be a higher degree of uncertainty for where a predicted intermediate point should be located between two rings further away from the lidar. Another contributing factor could be that the uncertainty in the real lidar measurements usually increases with increasing range. We can therefore expect that some of these variations between the predicted clouds and the original cloud also can be partly attributed to range dependent sensor noise.

In Figure 4.16 we can see the predicted point cloud of a wall from 32 and 64 channels to 128 channels, compared to the original point cloud. We can see that close to the left edge of the wall, the predicted point clouds are significantly sparser than the original cloud. We can also observe that the effect is slightly more noticeable in the predicted point cloud from 32 to 128 channels in Figure 4.16a than from 64 to 128 channels in Figure 4.16b. This effect is an expected result of the uncertainty noise filtering, since points that are located at or close to the edge, will likely be subject to the highest degree of smoothing in the super resolution network and hence they are more susceptible to the uncertainty filtering. Still, it seems like the predicted point clouds align relatively well with the original point cloud further away from the edge of the wall.

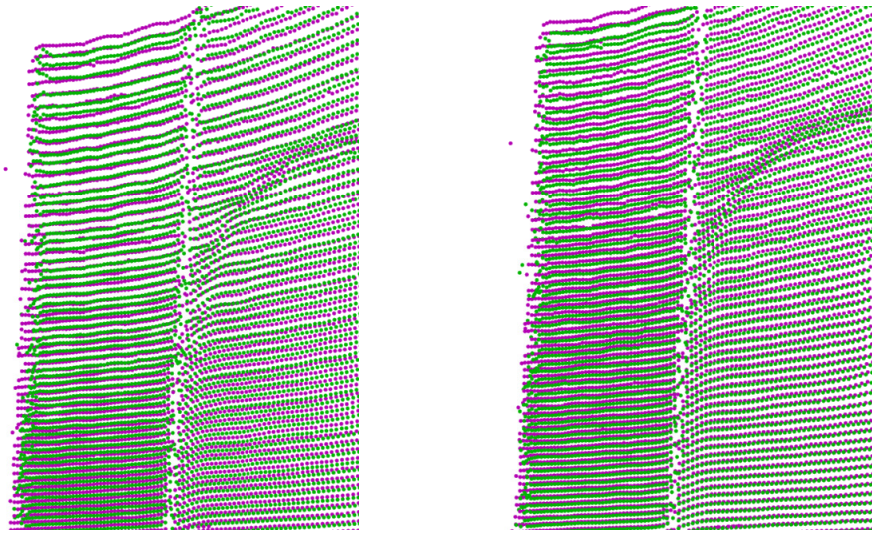


(a) Super resolution prediction from 32 to 128 channels



(b) Super resolution prediction from 64 to 128 channels

Figure 4.15: The original 128 channels point cloud (purple) from a bird's eye view, compared to the predicted point cloud (green) using super resolution from 32 to 128 channels (a) and from 64 to 128 channels (b). We can see for instance on the left side, that as we get further away from the point cloud center, the predicted point cloud from 64 to 128 channels is significantly closer to the original point cloud compared to the predicted point cloud from 32 to 128 channels.



(a) Super resolution from 32 to 128 channels **(b)** Super resolution from 64 to 128 channels

Figure 4.16: The original 128 channels point cloud (purple) of the edge of a wall, compared to the predicted point cloud (green) using super resolution from 32 to 128 channels (a) and from 64 to 128 channels (b). We can observe that there are slightly fewer points along the left edge of the wall in the predicted cloud from 32 to 128 channels than in predicted cloud from 64 to 128 channels.

Challenging Location

In Figure 4.17, a point cloud of the entrance to a building can be seen. An important detail is that the entrance is located slightly closer to the lidar than the rest of the building, which results in a relatively narrow horizontal gap in the point cloud at the top of the entrance. This is as mentioned in Section 3.11.3 a scenario which is likely to be challenging for the range super resolution network. The predicted point clouds from 32 to 128 channels and 64 to 128 channels of the same entrance can be seen in Figure 4.18.

We can observe that the predicted point clouds have noticeable artifacts in the point cloud gap, with free-floating intermediate points between the edges of the entrance and the rest of the building. This effect is especially noticeable in the predicted point cloud from 32 to 128 channels in Figure 4.18a. This result can most likely be explained by the fact that the building behind the entrance causes a less sharp discontinuity in the range image along these edges. As a consequence, it will be harder for the network to distinguish the entrance from the rest of the building and hence it becomes more challenging to remove these outliers using the uncertainty filtering. This undesirable effect could potentially be reduced by lowering the uncertainty threshold, so that more points are removed, but this would of course come at the cost of removing more points elsewhere in the point clouds as well.

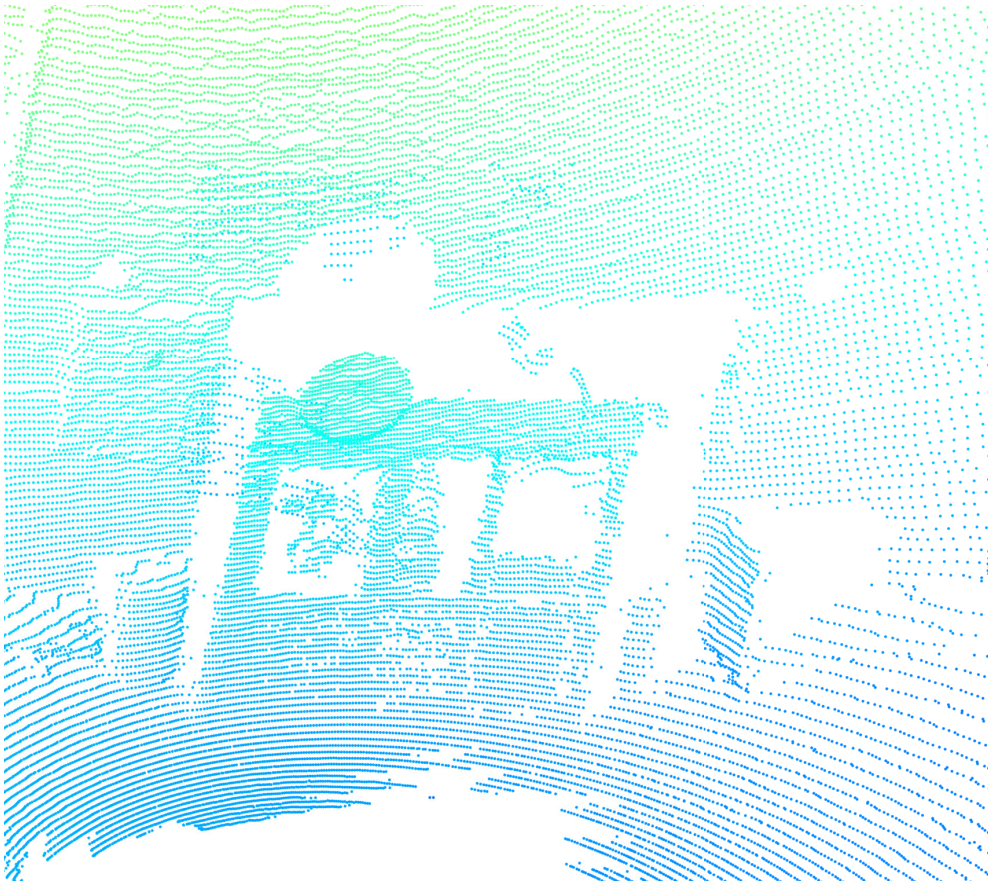
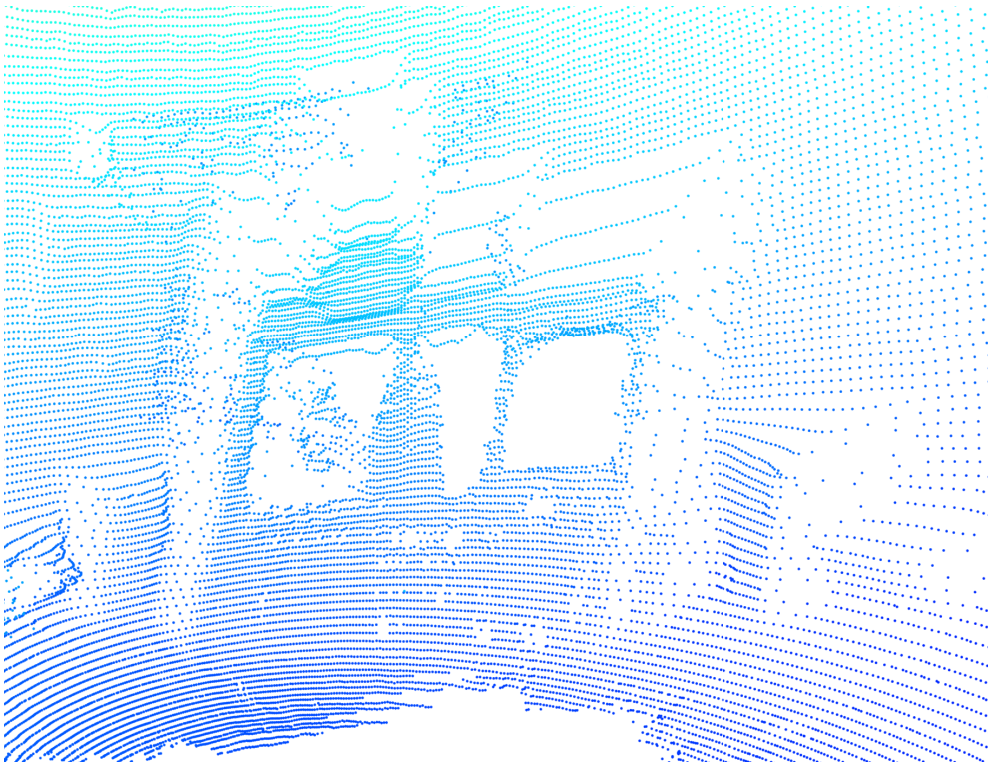
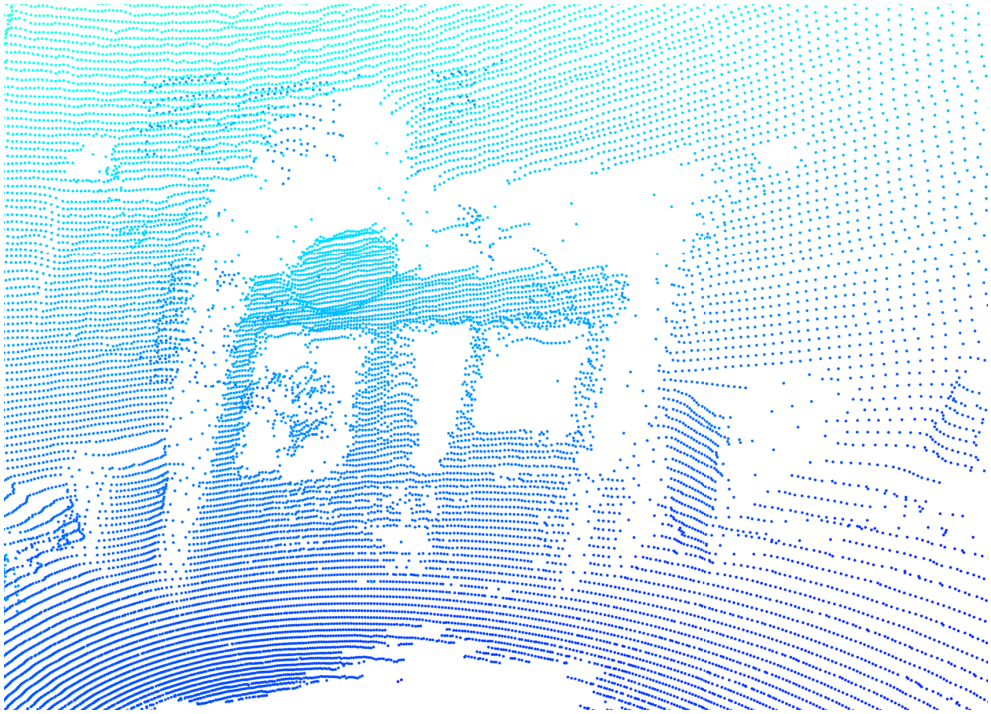


Figure 4.17: The original 128 channel point cloud of the entrance to a building. Notice that there is a relatively narrow gap in the point cloud along the edges of the entrance.



(a) Predicted point cloud from 32 to 128 channels.



(b) Predicted point cloud from 64 to 128 channels.

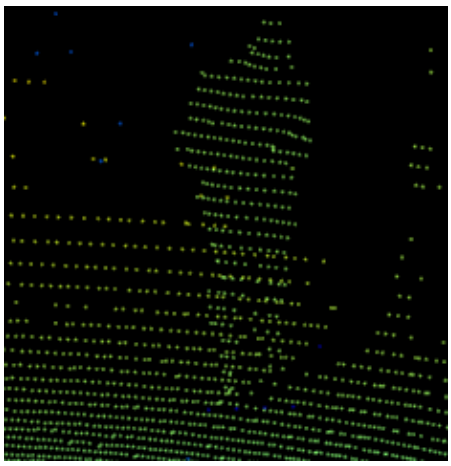
Figure 4.18: In (a) we see the predicted point cloud from 32 to 128 channels of the entrance to a building, while point cloud (b) is the predicted point cloud from 64 to 128 channels. We can see that there are significant artifacts along the edges of the entrance, which is particularly noticeable in the predicted point cloud from 32 to 128 channels.

Person Detection

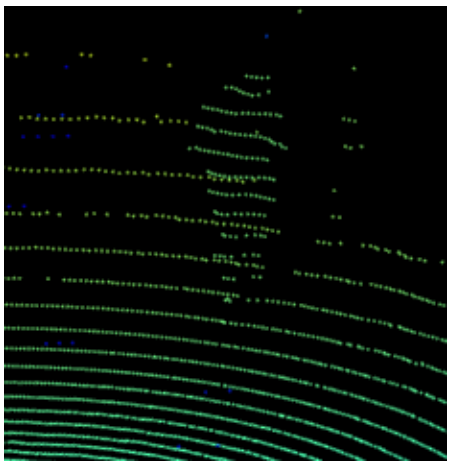
Here the results of the person detection evaluation test described in Section 3.11.3 are covered. In Figure 4.19 we can see the point cloud of a person at the full 128 lidar channel resolution, its downsampled counterpart to 32 and 64 channels and the corresponding predicted point clouds using super resolution. We can observe that in the point cloud that has been downsampled to 32 channels shown in Figure 4.19c, it is relatively difficult to determine that the point cloud originates from a walking human. If we compare this to the point cloud that has been upsampled from 32 to 128 channels using the super resolution network shown in Figure 4.19e, then it is considerably easier to see that the super-resolved point cloud is of a person.

This result can also partly be observed in the upsampled point cloud from 64 to 128 channels shown in Figure 4.19d compared to the downsampled 64 channels point cloud shown in Figure 4.19b. However, the effect is not as noticeable as in the previous case, since the 64 channels resolution seemingly is high enough to be able to differentiate the person from the rest of the point cloud. We can also observe that the prediction from 64 to 128 channels in Figure 4.19d preserves the contour of the person slightly better than the 32 to 128 channel prediction in Figure 4.19e.

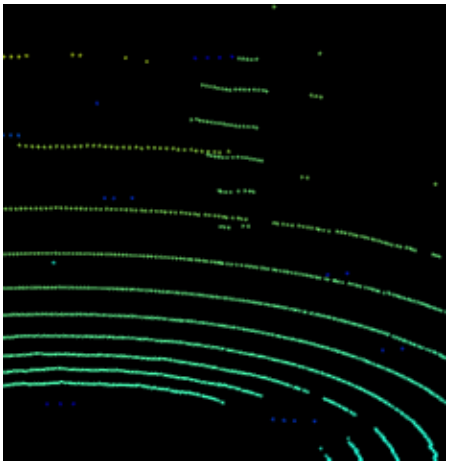
From a practical standpoint, it appears like the super resolution network from 32 to 128 channels could potentially make it easier for an object detection algorithm to detect the human compared with the low resolution 32 channels point cloud. However, in the predicted point cloud from 64 to 128 channels, it seems like the potential detection improvements are a bit more modest. There is also a potential trade-off that has to be made in terms of lower depth accuracy in the detection when using the super-resolved point cloud, compared to the original low resolution point cloud.



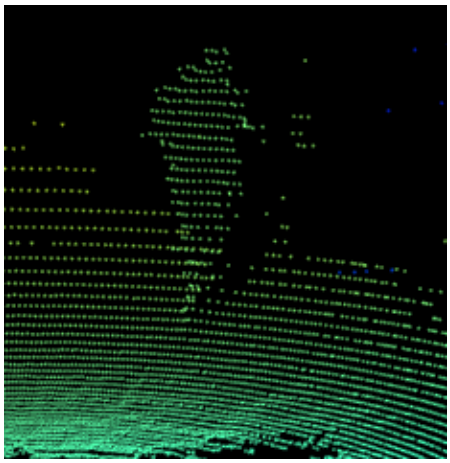
(a) High Resolution - 128 channels



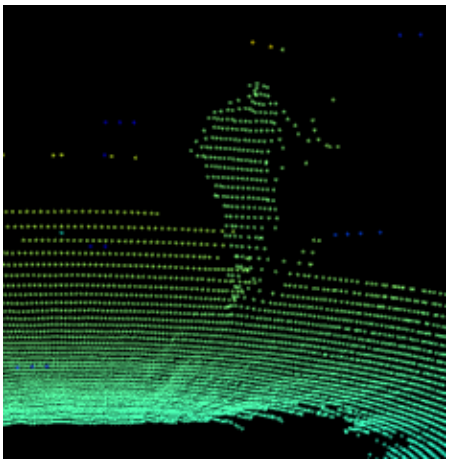
(b) Low Resolution - 64 channels



(c) Low Resolution - 32 channels



(d) Super Resolution from 64 to 128 channels



(e) Super Resolution from 32 to 128 channels

Figure 4.19: In (a) a person walking is observed at the full lidar resolution of 128 channels. The decimated low resolution images of the same person are seen in (b) and (c) at resolutions of 64 and 32 channels respectively. The resulting clouds using the super resolution network are shown from 64 to 128 channels in (d) and from 32 to 128 channels in (e). We can observe that it is difficult to see that it is a person when only having access to 32 channels as in (c). However, with the super resolution network from 32 to 128 channels, it is significantly easier to see that it is in fact a person.

4.2.2 Ambient and Intensity Super Resolution Results

Here the results of the intensity and ambient super resolution networks are presented and discussed, based on the evaluation methods described in Section 3.11.4. The results are presented as follows: First the quantitative results of super-resolving ambient and intensity images are covered. Then a qualitative assessment of the super-resolved intensity and ambient images is performed. Finally, we go through the results of coloring a 3D point cloud using predicted intensity images.

An overview of the PSNR and SSIM of the upscaled intensity and ambient images using the super resolution networks compared to bicubic interpolation are given in Table 4.3. The super resolution networks outperforms bicubic interpolation for these metrics in all cases, except in the case of the ambient image from 64 to 128 channels, where bicubic interpolation has slightly better results.

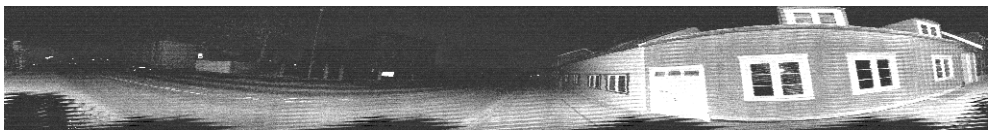
Table 4.3: Average PSNR [dB] / SSIM values (higher is better) for the super resolution network compared to bicubic interpolation on 16-bit intensity and ambient images.

	Intensity PSNR/SSIM		Ambient PSNR/SSIM	
Channel upscale	32 → 128	64 → 128	32 → 128	64 → 128
Super Resolution	**20.2/0.58**	**24.0/0.78**	**18.5/0.39**	20.5/0.57
Bicubic Interpolation	17.5/0.36	20.6/0.60	17.8/0.33	**20.7/0.59**

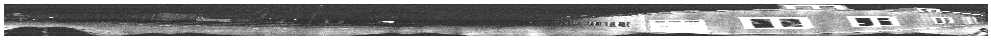
We can also see that the difference between super resolution and bicubic interpolation with respect to PSNR and PSNR is significantly larger for the intensity image compared to the ambient image. However, the PSNR value for 16-bit images is typically considered high when it is larger than 60 dB [119], while here all the PSNR values are less than 25 dB. Given that not only the super resolution networks, but also bicubic interpolation have extremely low PSNR values, this indicates that PSNR is not a reliable metric in this case. The most probable explanation for this, is that the assumption in the PSNR metric that the original high resolution image is noise free, is far from valid for real lidar images.

Qualitative Assessment - Intensity Image

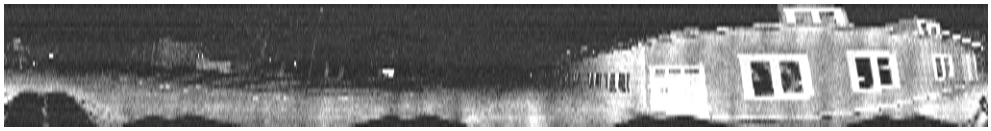
Figure 4.20 and Figure 4.21 shows the result of upscaling an intensity image from 32 to 128 channels and from 64 to 128 channels respectively, compared to bicubic interpolation. We can observe that the super-resolved intensity image from 32 to 128 channels in Figure 4.20d seems to be too smooth compared to the original image in Figure 4.20a. This is a common problem for many super resolution methods, when the low resolution image has been subject to severe aliasing due to high downsampling like here [120]. On the other hand, the predicted intensity image from 64 to 128 channels shown in Figure 4.21d looks sharper than the result of bicubic interpolation seen in Figure 4.21c.



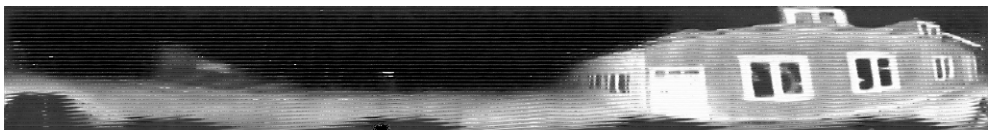
(a) High Resolution - 128 channels



(b) Low Resolution - 32 channels



(c) Bicubic interpolation from 32 to 128 channels

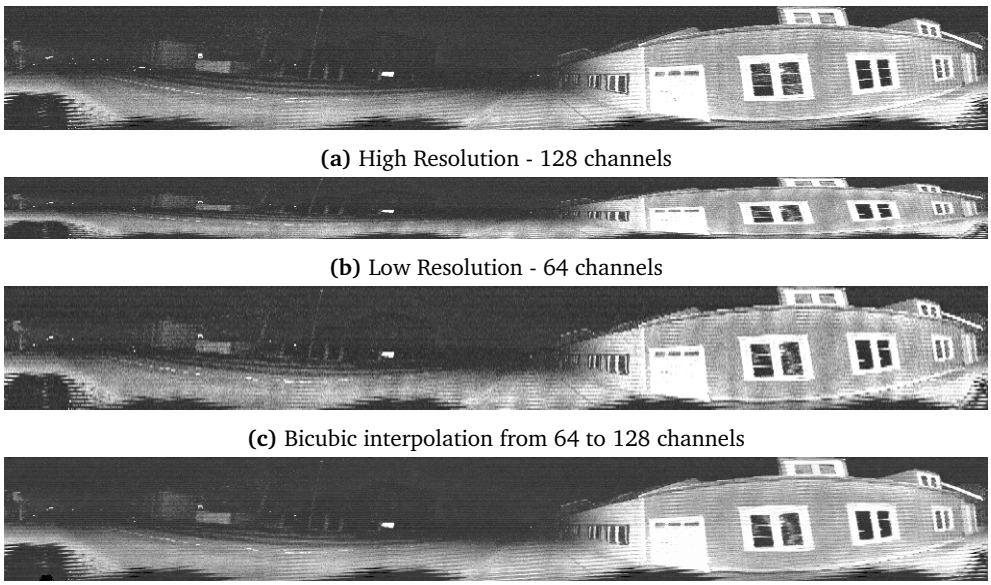


(d) Super Resolution from 32 to 128 channels

Figure 4.20: Comparison between super resolution and bicubic interpolation from 32 to 128 channels on a lidar intensity image. We can see that the super-resolved image (d) is a lot smoother than the result of bicubic interpolation (c). By inspection of the top edge of the building on the right, we see that bicubic interpolation results in a more uneven boundary than the super-resolved image. This indicates that super resolution network might preserve the structure in the image better.

By close inspection of the edge along the top of the building in Figures 4.20 and 4.21, it appears like the intensity super resolution network images preserve the edge boundary slightly better than bicubic interpolation. This result is also supported by the higher reported SSIM in Table 4.3, since higher SSIM usually corresponds to a better preservation of the underlying structure [86]. Better edge preservation could be useful if the super resolution networks should be used in e.g. object segmentation. Though, it is not easy to determine if the super-resolved image from 32 to 128 channels shown in Figure 4.20d is any better (or worse) than applying standard bicubic interpolation for this purpose. However, the prediction from 64 to 128 channels in Figure 4.21d seems to resemble the original high resolution image a bit better than bicubic interpolation in Figure 4.21c, although the difference is not that large. It is likely that this small difference is not worth the extra computational complexity of using the super resolution network compared to standard bicubic interpolation for most purposes.

A likely explanation for why super resolution does not yield better results here, is that the original high resolution image contains a relatively high degree of noise, which was also indicated by the low PSNR values in Table 4.3. Since these images were used as ground truth during training, this likely made it challenging for the



(a) High Resolution - 128 channels

(b) Low Resolution - 64 channels

(c) Bicubic interpolation from 64 to 128 channels

(d) Super Resolution from 64 to 128 channels

Figure 4.21: Comparison between super resolution and bicubic interpolation from 64 to 128 channels for a lidar intensity image. We can observe that the super-resolved image (d) has less "jagged" edges along the top of the building than bicubic interpolation (c). The super-resolved image also seems sharper than the image from bicubic interpolation.

network to converge to a better solution. Applying super resolution methods on non-synthetic images is known to be very challenging [121], so this was not an unexpected result.

We can also observe that both in the predicted images from 32 to 128 channels in Figure 4.20d and from 64 to 128 channels in Figure 4.21d, there is a noticeable repeating horizontal line pattern. The effect is most noticeable in the darker regions in the predicted intensity image from 32 to 128 channels in Figure 4.20d. This result can almost certainly be explained by the fact that these lines correspond to the same channels as those that are available in the low resolution image and hence the network will likely have a higher confidence in these predictions. One method that would likely have reduced this effect, is to add blurring in the degradation model when creating the low resolution images from the full 128 channel intensity images. However, this degradation model is not realistic in the sense that the only difference between the images should be the number of measured lines, since we are only simulating a reduction in the channel number. If we introduce blurring into the degradation model, then we have essentially applied a low pass filter to the high resolution images, which would reduce the amount of aliasing in the downsampling process [84]. Although this would perhaps yield a more visually pleasant result here, the final model would not be of any practical

use, since the degradation model would not reflect reality.

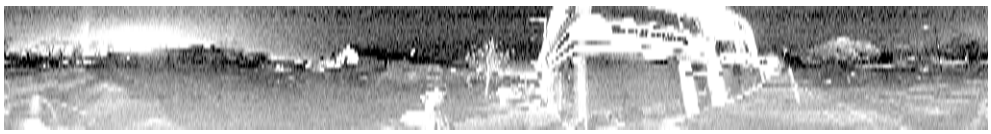
Qualitative Assessment - Ambient Image

In Figure 4.22 and Figure 4.23, we see the result of upscaling an ambient image from 32 to 128 channels and from 64 to 128 channels respectively. We can observe that the super-resolved images appear to be overly smoothed compared to the original image shown in Figure 4.22a. This smoothing artifact is a common challenge when applying SISR methods to real images [81], and here we see that the there is an undeniable reduction in sharpness in the super-resolved images. As we can see, the original high resolution image shown in Figure 4.22a is relatively grainy. A likely explanation for this result is therefore that the ambient images contain too much noise for the super resolution network to be able to learn a mapping without these smoothing artifacts. It was also observed during the training phase that it was significantly more challenging to make the ambient super resolution network converge to any meaningful result relative to the intensity super resolution network.

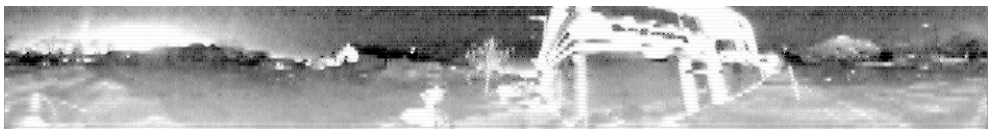
Based on the results we see here, it seems unlikely that the ambient super resolution network is better than just applying bicubic interpolation. Improving the results of the super-resolved ambient images would most probably require a more rigorous handling of the high degree of noise in the original ambient images. One possible approach could be to take multiple images at the exact same location and computing the average or median pixel value to generate the high resolution image, while the low resolution image could be generated based on only one of the images. This could potentially make the training more stable, since the high resolution image would contain less white noise, but it would require that the environment is completely static and that the lidar is not in motion when sampling the images at a given location.

(a) High resolution - 128 channels

(b) Low resolution - 32 channels

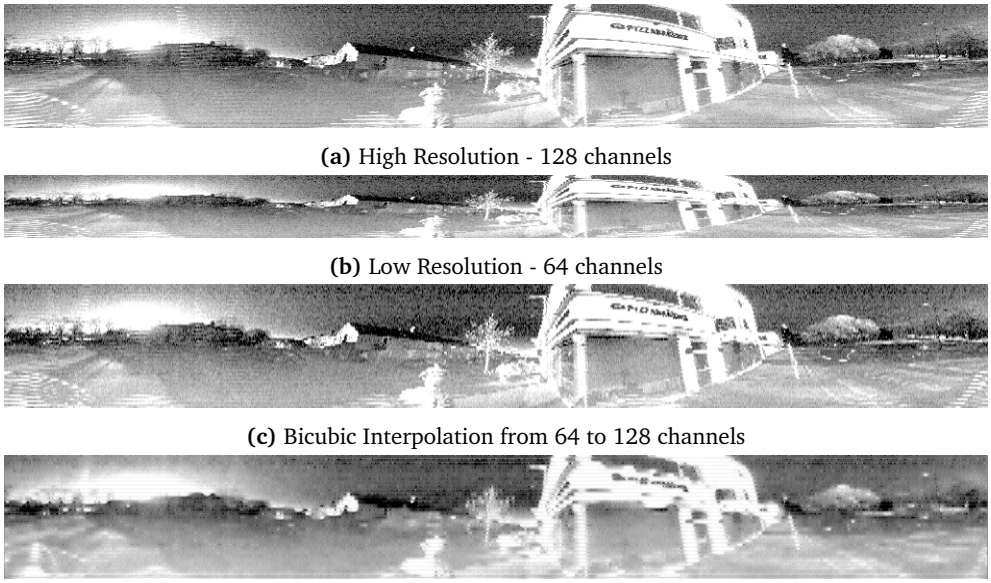


(c) Bicubic interpolation from 32 to 128 channels



(d) Super resolution from 32 to 128 channels

Figure 4.22: Comparison between the original 128 channel ambient high resolution image (a), downsampled 32 channel low resolution image (b), upsampled image with bicubic interpolation (c) and the upsampled image using super resolution (d). We can see that the bicubic interpolation results in a grainier image compared to using super resolution, but this comes at a cost of a reduction in the perceived sharpness in the super resolution image.



(a) High Resolution - 128 channels

(b) Low Resolution - 64 channels

(c) Bicubic Interpolation from 64 to 128 channels

(d) Super resolution from 64 to 128 channels

Figure 4.23: Comparison between the original 128 channel ambient high resolution image (a), downsampled 64 channel low resolution image (b), upsampled image with bicubic interpolation (c) and the upsampled image using super resolution (d). We can see that bicubic interpolation results in a grainier image compared to using super resolution, but this comes at a cost of a reduction in sharpness in the super resolution image.

Intensity Colored Point Cloud Results

Here the results of coloring a 3D range point cloud with predicted intensity images as described in Section 3.11.4, are covered. In Figure 4.24 we can see a 3D range point cloud from the lidar super resolution test set described in Section 3.9.2, which has been colorized using the original intensity image from the same location.

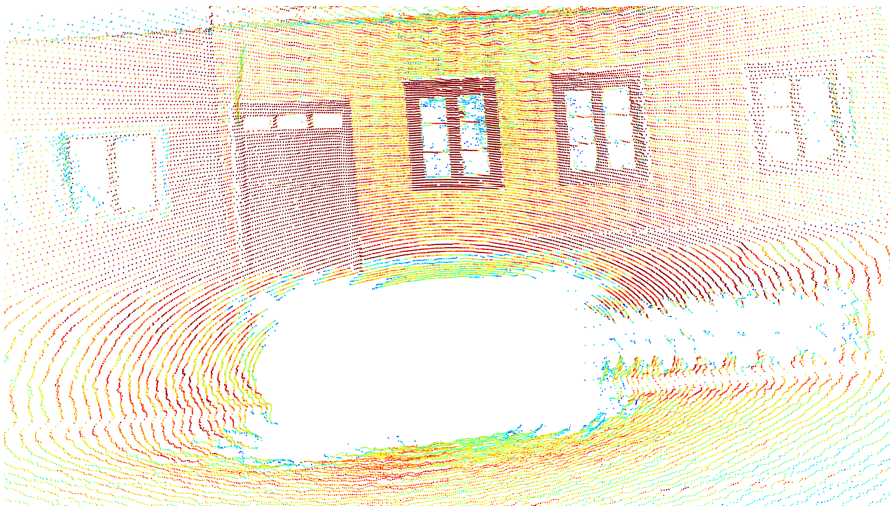
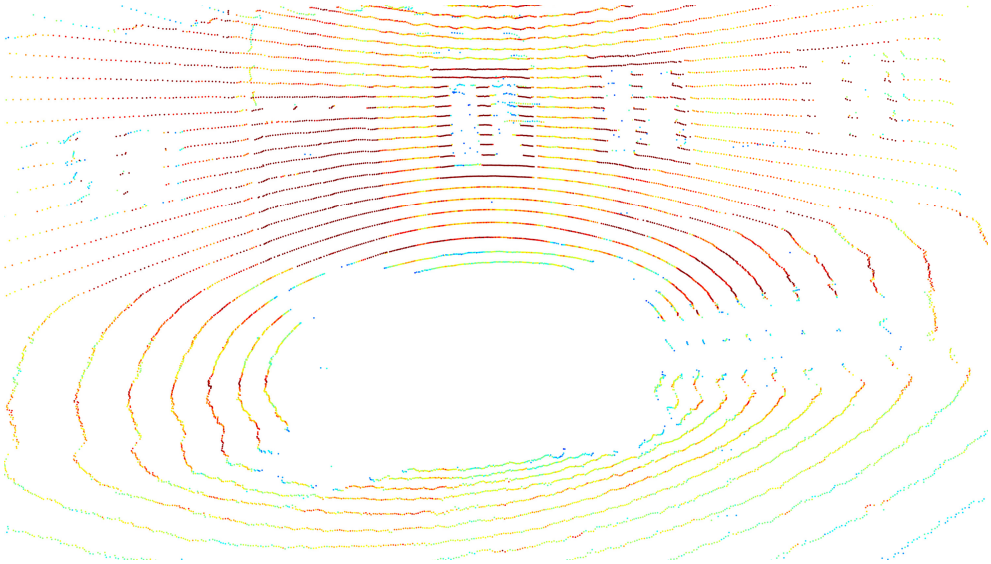
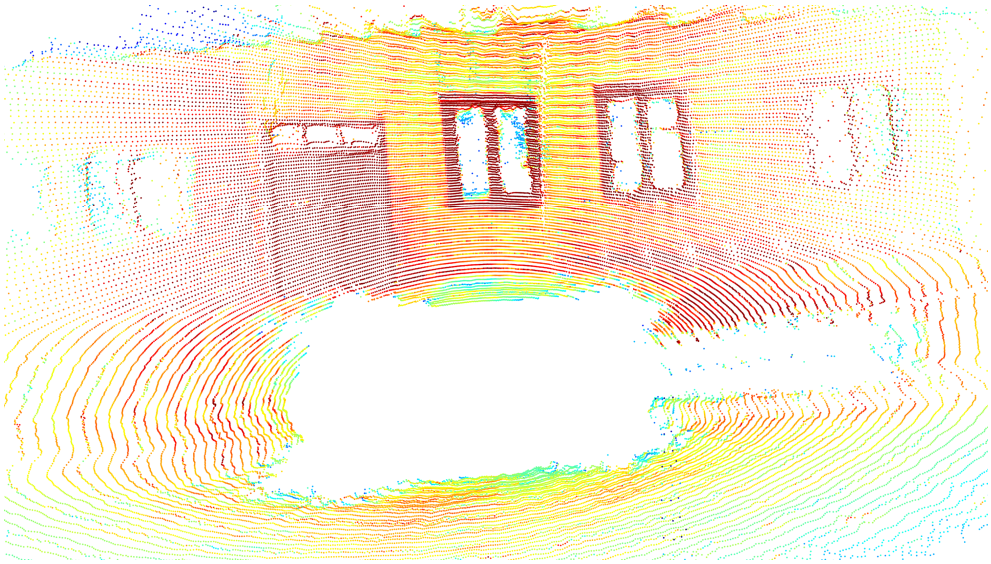


Figure 4.24: A 128 channel 3D range point cloud that has been colorized using the intensity image at the same location. We can see that the intensity coloring results in the door and windows to become more noticeable.

In Figure 4.25 and Figure 4.26, the predicted point cloud using the range super resolution network, colorized with the predicted intensity image from 32 to 128 channels and 64 to 128 channels respectively, at the same location as the point cloud in Figure 4.24, are shown. We can see that the intensity coloring makes it relatively easy to distinguish some structures such as doors and windows from the rest of the point cloud. We can also see that both the predicted point cloud and intensity from 32 to to 128 channels in Figure 4.25b and from 64 to 128 channels in Figure 4.25b looks relatively similar to the original point cloud seen in Figure 4.24. This shows a potential use case for combining the intensity super resolution network with the range super resolution network for i.e., object detection or dense mapping. One limitation of this method, is that the Ouster intensity image does not correspond to the raw intensity information from the point cloud. Usually, it is the raw intensity information that is used in combination with a point cloud and not the intensity information after post-processing as we see here. This limitation could possibly be overcome by training a network on raw intensity images instead of the post-processed intensity images. The author of this thesis did also attempt this, but it proved to be challenging to make the training converge to sensible results, so it would likely require a network architecture that is designed for raw images.

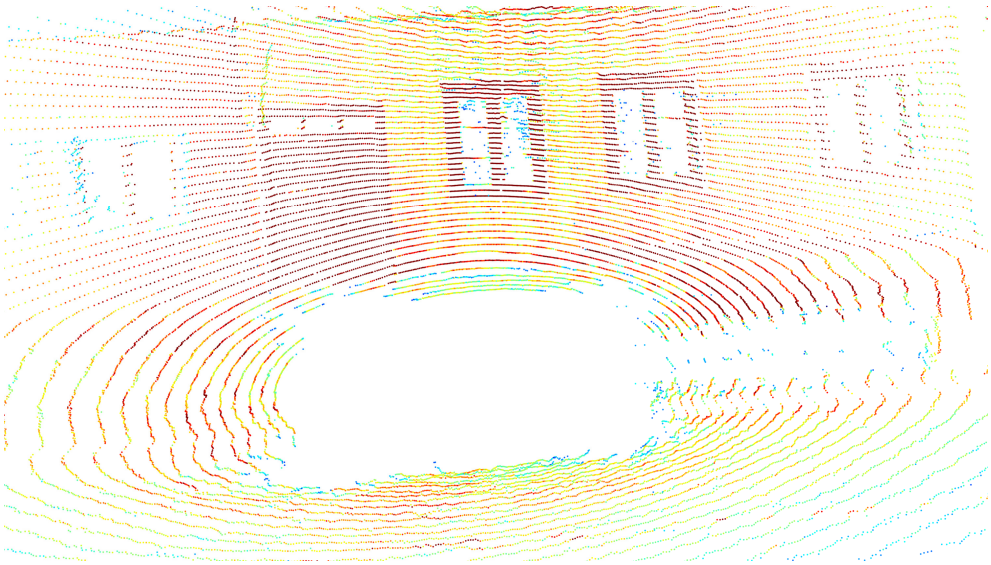


(a) Low resolution - 32 channels

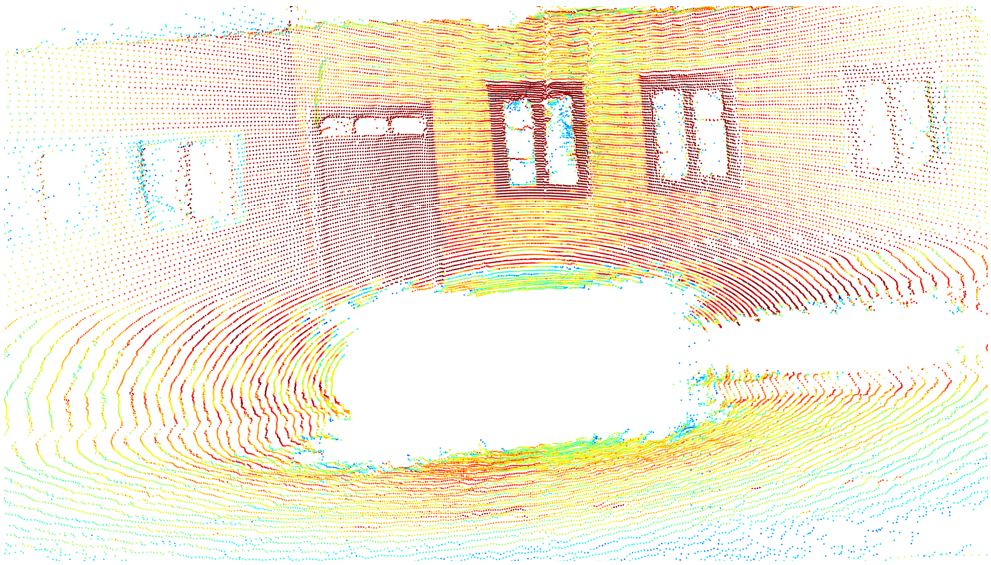


(b) Super resolution from 32 to 128 channels

Figure 4.25: A point cloud colored with intensity that has been downscaled from 128 to 32 channels (a) and the super-resolved 128 channel point cloud with intensity (b). We can see that although there are some clearly visible artifacts in the super-resolved point cloud like bent edges, the super resolution network seems to be able to reasonably densify the point cloud given how sparse the original point cloud is.



(a) Low resolution - 64 channels

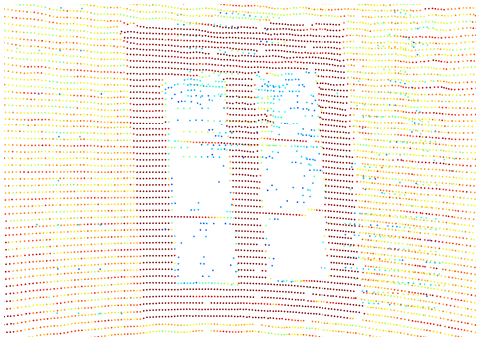


(b) Super resolution from 64 to 128 channels

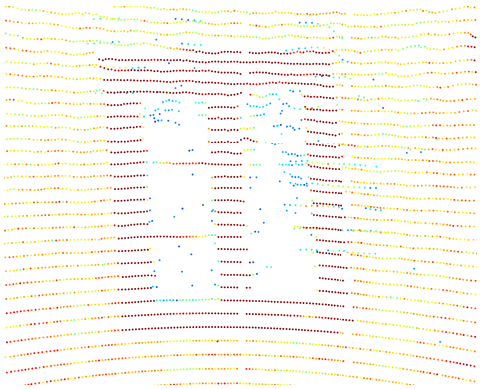
Figure 4.26: A point cloud colored with intensity that has been downscaled from 128 to 64 channels (a) and the super-resolved 128 channel point cloud colored with intensity (b).

In Figure 4.27 a closer view of one of the windows in the point cloud from Figures 4.24 to 4.26, is shown. We can observe that the original point cloud seen in Figure 4.27a contains significant noise, which is likely caused by the transparent and reflective properties of the glass in the window, which is a common challenge faced when using lidars [122]. However, most of the noisy points can from a human perspective be relatively easily be differentiated from the rest of the window, based on the colorization from the intensity mapping. Similarly, by inspecting the super-resolved point clouds shown in Figures 4.27d and 4.27e, we see that it is possible to differentiate most of the window frame from the noise based on the distinctive red intensity color of the window frame.

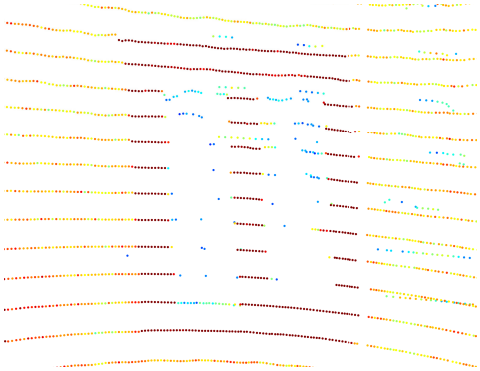
We can also observe that the narrow horizontal red bars in the window frame are completely removed in the point cloud downsampled to 32 channels in Figure 4.27c and partially removed in the point cloud downsampled to 64 channels in Figure 4.27b. As a result, the super-resolved point clouds are unsurprisingly not able to recover these horizontal bars. However, the wider regions of the window frame appears to be mostly recovered.



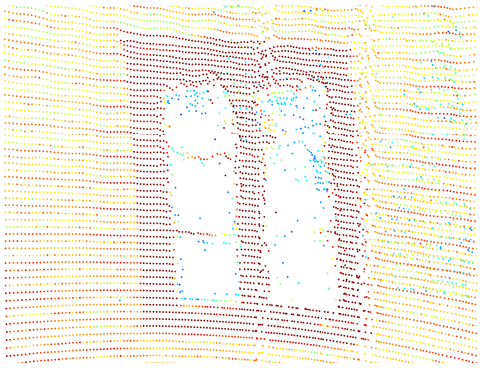
(a) High Resolution - 128 channels



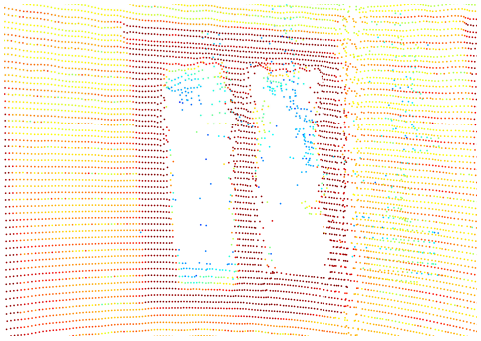
(b) Low Resolution - 64 channels



(c) Low Resolution - 32 channels



(d) Super Resolution from 64 to 128 channels



(e) Super Resolution from 32 to 128 channels

Figure 4.27: The original point cloud of a window colored by intensity (a), its downscaled version to 64 (b) and 32 (c) channels, the super-resolved point clouds from 64 to 128 channels (d) and from 32 to 128 channels (e). We can see that there is significant noise in the original point cloud, which also affects the predicted point clouds. We can also observe that the super resolution network has problems with predicting the horizontal bars in the window frame, since most of this is lost in the downsampling.

4.3 High Resolution Image Generation Results

In this section we cover the results of generating high resolution lidar images based on multiple point clouds as described in Section 3.12. The section is divided into two parts. First, in Section 4.3.1 we present the result of generating high resolution images based on point clouds from the Newer College dataset by Zhang et al. [113] as described in Section 3.12.1. Then, in Section 4.3.2 we go through the result of generating high resolution images based on point clouds from a self-collected dataset from an indoor entrance hall as described in Section 3.12.2.

4.3.1 Outdoor Images - Newer College

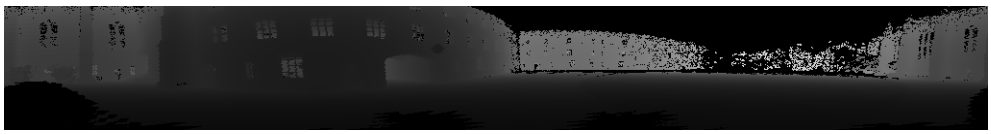
In Figure 4.28, 4.29 and 4.30, we see the original range, intensity and ambient image respectively from the same time instance in the Newer College dataset [113], together with the constructed image with twice as high vertical resolution, using the pipeline described in Section 3.7.

Range Image

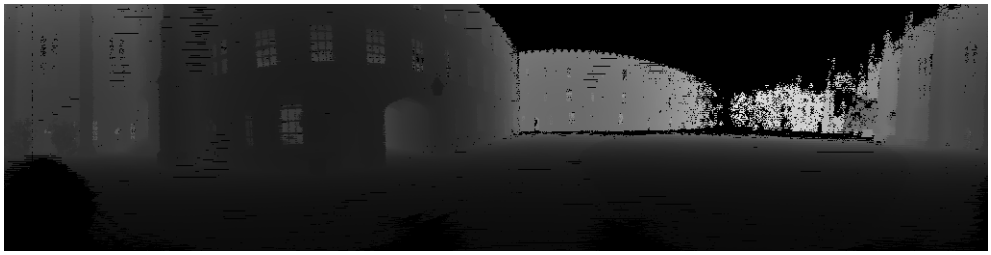
We can observe that the constructed range image shown in Figure 4.28b seems to be relatively similar to the original range image shown in Figure 4.28a, though the constructed image contains a few noticeable black stripes due to missing range measurements. These lines could possibly be removed by applying a hole filling algorithm, but this could result in either inaccurate depth values at these locations or potentially filling regions in the image where there in fact are no objects. A more optimal solution would be to include more point clouds in the construction of the image. However, since the Newer College dataset is collected while in motion [113], this can result in artifacts where some points are included in the constructed image, which are not visible from the location of the original low resolution image due to occlusion or limited range. This is naturally not ideal, since this would not be a realistic simulation of a lidar with a higher number of channels.

Intensity Image

In Figure 4.29a we see an original lidar intensity image together with a constructed intensity image with twice the vertical resolution. One difference we can observe immediately between the images, is that in the regions where there are no available range measurements, e.g. in the sky, the constructed image is completely black. This challenge could possibly be dealt with if these images are used to train a super resolution network by only training on patches of the image where there are available measurements. A more optimal solution would perhaps be to limit the application of the image generation pipeline to confined spaces, to reduce the number of missing measurements. In the regions where there are available measurements, the images appear similar, but there is a noticeable reduction in white



(a) Original range image with 128 channels.



(b) Constructed range image with 256 channels.

Figure 4.28: Comparison between the original range image (a) with 128 channels and the constructed range image (b) with 256 channels. Note that the images have been re-scaled to fill the entire 16-bit image resolution for visualization purposes. We can observe that although the images look relatively similar, there are horizontal lines with missing data in the constructed image. Data from the Newer College dataset [113].

noise in the constructed image. This comes as a result of using the median intensity value of the points that map to the same pixel in the image. Having access to a less noisy high resolution image than the original intensity images, could possibly make the training phase of a super resolution network more stable, though this will require further testing.

We can observe that the hole filling algorithm described in Section 3.7.5 seems to give reasonable results, since it is relatively hard to notice that the lines of missing measurements that we saw in the constructed range image in Figure 4.28b, have been filled in the constructed intensity image in Figure 4.29b. However, there are some clear artifacts of the hole filling algorithm along the edges of the black regions at the bottom of the image in Figure 4.29b, though these regions could be excluded by only training on patches of the image.

Another observation that can be made for Figure 4.29, is that the left edge of the arc-shaped entrance in the center of the image is a bit more jagged in the constructed image than in the original intensity image. This is an expected result due to inaccuracies in the point cloud registration and discretization errors that are introduced in the point cloud projection. Another likely source of these inaccuracies, is that the point clouds are partially distorted, since the point clouds are sampled while the lidar is in motion.

Based on the constructed intensity image that we see here, it could be possible to train a super resolution network if we have access to enough images. However, there are some limitations to this approach. First, it is a lot more time consuming to generate a training set with these images compared to collecting the original intensity images, partly because we need to ensure the point cloud registration has

(a) Original intensity image with 128 channels.

(b) Constructed intensity image with 256 channels.

Figure 4.29: Comparison between the original intensity image (a) with 128 channels and the constructed intensity image (b) with 256 channels. We see that except for a few artifacts around the edges at the bottom, resulting from the hole filling algorithm, the images do look similar. Since the constructed image requires range measurements in order to fill a pixel, the sky is completely black. This suggests that the algorithm may be more suited for confined spaces. Data from the Newer College dataset [113].

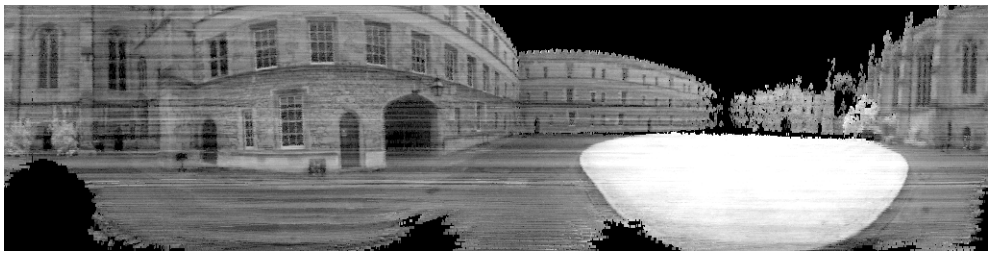
converged to a valid solution for every image we create. It will also require that the environment we collect the data in is completely static, since moving objects like cars, will naturally be detrimental to the image generation process. On the other hand, an advantage with generating intensity images by this pipeline, is that rendering realistic intensity images using computer simulation is very challenging relative to e.g. the range images that we saw in Figure 4.28 [110]. However, it will require further research to know if the inevitable inaccuracies introduced in the image generation process are small enough that a super resolution network trained on these images can be useful.

Ambient Image

Figure 4.30 shows an ambient lidar image from the Newer College dataset [113], together with the constructed ambient image. We can see that the constructed ambient image shown in Figure 4.30b contains some very noticeable white stripes, that are not present in the original image shown in Figure 4.30a. Based on the result, it seems like that there might be too much noise in the ambient data for the constructed image to look realistic. The black pixels due to missing range measurements are also clearly more noticeable here, than what we saw for the intensity image in Figure 4.29b. It seems unlikely that these constructed ambient images can be used to train a super resolution network, since they contain a high degree of noise and do evidently not represent the original images very well. Based on these observations, it might be better to limit the application of the image

generation pipeline to range and intensity images.

(a) Original ambient image with 128 channels.



(b) Constructed ambient image with 256 channels.

Figure 4.30: Comparison between the original ambient image (a) with 128 channels and the constructed ambient image (b) with 256 channels. We can see that the constructed image is noisy with noticeable white stripes across the image, so it does not represent the original image very well. Data from the Newer College dataset [113].

4.3.2 Indoor Images - Entrance Hall

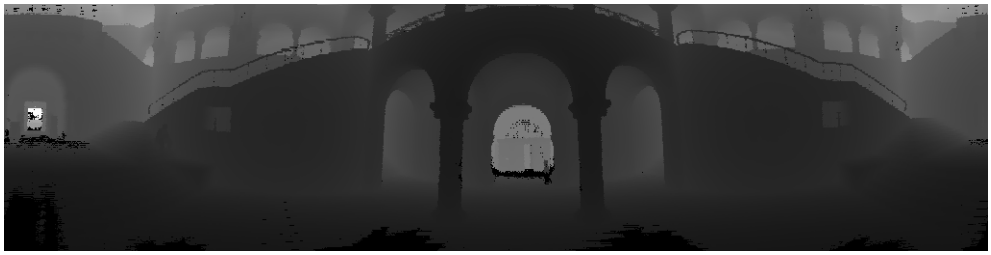
In Figure 4.31 and Figure 4.32 we see the original range and intensity images from an indoor location compared with the same type of images constructed by the image generation pipeline, as described in Section 3.12.2. Note that the ambient image is not of interest here, since we are in an indoor environment, where there is little near-infrared ambient lighting.

Range Image

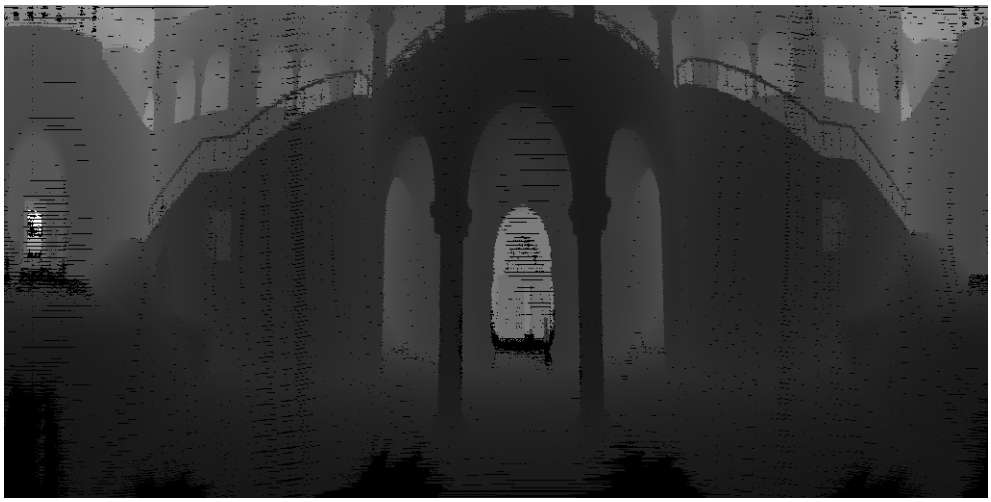
We can observe that the constructed range image in Figure 4.31b with twice as high vertical resolution compared to the original range image in Figure 4.31a, has significantly fewer black pixels due to missing measurements. This is a clear improvement from the constructed range image in Figure 4.28b, which was based on point clouds from the Newer College dataset. This is of course mainly a result of having access to more point clouds from the same location captured with minimal motion and that the point clouds are collected in a smaller environment. We can see that in the constructed range image in Figure 4.31c with four times higher vertical resolution than the original image in Figure 4.31a, the black lines resulting from missing measurements are starting to dominate the image. This problem could most likely be mitigated by rotating the lidar slower and including more point clouds in the image generation. Based on these observations, it seems like the double vertical upscaling seen in Figure 4.31b is the limit for what could

potentially be used to train a super resolution network with the data collected here.

(a) Original range image with 128 channels.



(b) Constructed range image with 256 channels.



(c) Constructed range image with 512 channels.

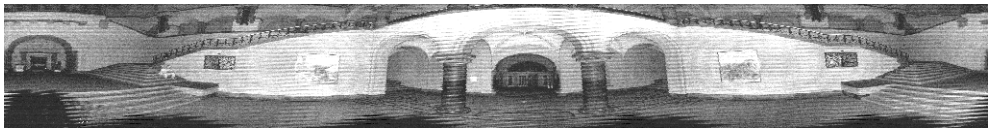
Figure 4.31: Comparison between the original range image (a) and two constructed range images with a vertical resolution that is 2× higher (b) and 4× higher (c) than the number of channels of the lidar. We can see that the constructed image with 512 channels (c) appears to have too many missing range measurements, while the constructed image with 256 channels (b) looks more reasonable.

Intensity Image

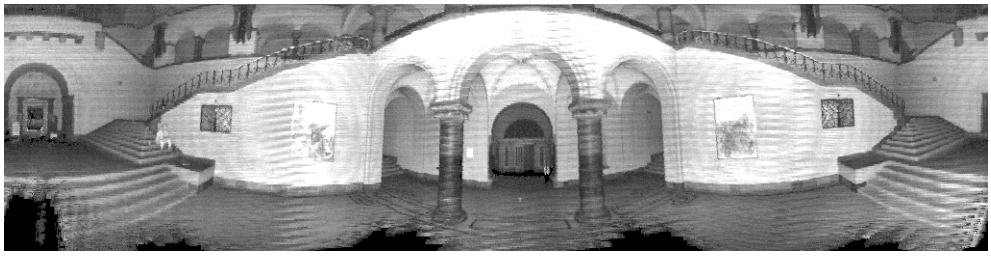
In Figure 4.32 we see the original intensity image from the same location as in Figure 4.31, together with the constructed intensity images with two times and four times higher vertical resolution. We can observe that the intensity image in Figure 4.32b is considerably sharper compared to the original intensity image in Figure 4.32a. The constructed image with four times higher vertical resolution in

Figure 4.32c does also seem to contain less white noise than the original intensity image, but there is a noticeable degradation in the image quality from what we see in Figure 4.32b. This is also in line with what we would expect given the number of missing measurements that we saw in the range image from the same location in Figure 4.31c.

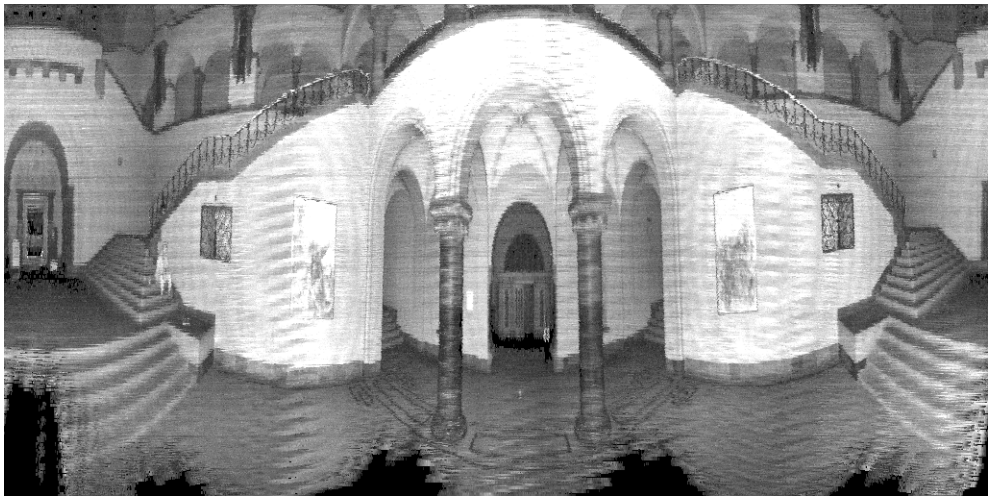
Another important observation, is that since this is an enclosed space, the constructed images look significantly more realistic compared to what we saw in the constructed intensity images from the Newer College dataset in Figure 4.29, as we do no longer have a completely black sky. Although it is not possible to say if these images can be used to train a super resolution network, the overall reduction in white noise seems promising, given that this was a major challenge when training the super resolution networks directly on the original lidar images. However, creating a large enough dataset that it could be used to train a super resolution network, will obviously be very time consuming. Due to time constraints, it was unfeasible to create a large enough dataset using the image generation pipeline described in Section 3.7, to test if a super resolution network can be trained using these images. This will therefore require further research.



(a) Original intensity image with 128 channels.



(b) Constructed intensity image with 256 channels.



(c) Constructed intensity image with 512 channels.

Figure 4.32: Comparison between the original intensity image (a) and two constructed range images with a vertical resolution that is 2× higher (b) and 4× higher (c) than the number of channels of the lidar. The constructed images looks relatively similar to the intensity image. We can observe that the constructed images contain less white noise than the original intensity image, though the constructed image with 512 channels in (c) seems to have a reduction in sharpness from (b), due to missing measurements.

Chapter 5

Conclusion

This master thesis has focused on two different research topics related to robotic perception. The first of these was to attempt to leverage both intensity and ambient lidar images in combination for improved robotic place recognition. The result of the work from this master thesis was a proof-of-concept visual place recognition method using lidar intensity and ambient images, which achieved 90% precision in an urban dataset spanning 4 km. By qualitative evaluation of the method, it was demonstrated that the intensity and ambient images can be utilized in a complementary manner to increase the robustness to perceptual aliasing occurring in one of the images.

The second research topic was to explore applying neural networks to increase the apparent resolution of the range, intensity and ambient images from a real lidar. A pre-existing lidar super resolution pipeline, which was originally designed only for lidar range data, was tested to see if it could be used to super-resolve each of the three individual lidar images. The tests indicate that the super resolution pipeline can be applied to real lidar range images and yield reasonable super-resolved results. Applying the super resolution pipeline to intensity and ambient images gave mixed results. It seems like the super-resolved intensity might be slightly better than bicubic interpolation, though this is not certain. Applying the super resolution pipeline to ambient images appears to result in worse performance than bicubic interpolation. The main problem is likely that the intensity images, and especially the ambient images, contain too much noise, which hinders the super resolution networks from training effectively.

A pipeline that can generate range, intensity and ambient lidar images based on multiple lidar point clouds, with a higher vertical resolution than the number of channels of the given lidar, has also been proposed. The results indicates that the pipeline can be used to generate realistic looking range and intensity images in enclosed environments. Generating ambient images did in contrast not yield satisfactory results. The generated intensity images appear to have a significant reduction in noise compared to the original intensity images and could therefore potentially be used to train an improved intensity super resolution network, though this will require further research.

As part of the thesis, a multi-modal handheld sensor rig has been built, tested and used for data collection. This sensor rig will hopefully be a useful contribution to future research in robotic perception.

5.1 Future Work

Some recommendations for future work are the following:

- Incorporating the proposed visual place recognition algorithm into a full-scale SLAM pipeline.
- Test if the trained range super resolution network can be used for dense mapping applications and/or for improved object detection.
- Generate a full intensity training set using the proposed image generation pipeline, to see if it can be used to improve the intensity super resolution network.
- Testing other super resolution architectures that can potentially work better on lidar images.
- Training on raw intensity images instead of post-processed intensity images.

Bibliography

[1] A. G. Kashani, M. Olsen, C. Parrish and N. Wilson, 'A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration,' *Sensors*, vol. 15, pp. 28 099–28 128, Nov. 2015. DOI: 10.3390/s151128099.

[2] Ouster, *OS0*, https://ouster.com/products/scanning-lidar/os0-sensor/. (visited on 12/04/2022).

[3] A. Pacala. (2018). 'Lidar as a camera – digital lidar's implications for computer vision,' [Online]. Available: https://ouster.com/blog/the-camera-is-in-the-lidar/ (visited on 27/05/2022).

[4] H. Yu, D. Liu, H. Shi, Y. Hanchao, Z. Wang, X. Wang, B. Cross and M. Bramlet, 'Computed tomography super-resolution using convolutional neural networks,' Sep. 2017. DOI: 10.1109/ICIP.2017.8297022.

[5] P. Shamsolmoali, M. Zareapoor, D. Jain, V. Jain and J. Yang, 'Deep convolution network for surveillance records super-resolution,' *Multimedia Tools and Applications*, vol. 78, pp. 1–15, Sep. 2019. DOI: 10.1007/s11042-018-5915-7.

[6] T. Yang, Y. Li, C. Zhao, D. Yao, G. Chen, L. Sun, T. Krajnik and Z. Yan, *3d tof lidar in mobile robotics: A review*, 2022. DOI: 10.48550/ARXIV.2202.11025. [Online]. Available: https://arxiv.org/abs/2202.11025.

[7] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick and A.-a. Agha-Mohammadi, 'Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion,' in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2518–2525. DOI: 10.1109/IROS45743.2020.9341361.

[8] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. De Boer, T. Koolen and J. Pratt, 'Team ihmc's lessons learned from the darpa robotics challenge trials,' *Journal of Field Robotics*, vol. 32, Mar. 2015. DOI: 10.1002/rob.21571.

[9] T. Gee, J. James, W. Van Der Mark, P. Delmas and G. Gimel'farb, 'Lidar guided stereo simultaneous localization and mapping (slam) for uav outdoor 3-d scene reconstruction,' in *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2016, pp. 1–6. DOI: 10.1109/IVCNZ.2016.7804433.

[10] M. Pierzchała, R. Astrup and P. Giguère, 'Mapping forests using an unmanned ground vehicle with 3d lidar and graph-slam,' *Computers and Electronics in Agriculture*, vol. 145, Feb. 2018. DOI: 10.1016/j.compag.2017.12.034.

[11] Slamtec, *Rplidar a1*, https://www.slamtec.com/en/Lidar/A1Spec. (visited on 07/05/2022).

[12] Neabot, *Neabot nomo q11 robot vacuum cleaner*, https://neabot.com/products/neabot-nomo-q11-smart-robot-vacuum. (visited on 07/05/2022).

[13] S. Royo and M. Ballesta-Garcia, 'An overview of lidar imaging systems for autonomous vehicles,' *Applied Sciences*, vol. 9, p. 4093, Sep. 2019. DOI: 10.3390/app9194093.

[14] D. L. Lu, *Ambient and reflectivity in ouster lidar*, https://github.com/ouster-lidar/ouster_example/issues/177. (visited on 06/04/2022).

[15] S. Lowry, N. Sünderhauf, P. Newman, J. Leonard, D. Cox, P. Corke and M. Milford, 'Visual place recognition: A survey,' *IEEE Transactions on Robotics*, pp. 1–19, Nov. 2015. DOI: 10.1109/TRO.2015.2496823.

[16] C. Stachniss, J. J. Leonard and S. Thrun, 'Simultaneous localization and mapping,' in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Cham: Springer International Publishing, 2016, pp. 1153–1176, ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_46. [Online]. Available: https://doi.org/10.1007/978-3-319-32552-1_46.

[17] K. Yousif, A. Bab-Hadiashar and R. Hoseinnezhad, 'An overview to visual odometry and visual slam: Applications to mobile robotics,' *Intelligent Industrial Systems*, vol. 1, Nov. 2015. DOI: 10.1007/s40903-015-0032-7.

[18] M. A. Uy and G. H. Lee, *Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition*, 2018. DOI: 10.48550/ARXIV.1804.03492. [Online]. Available: https://arxiv.org/abs/1804.03492.

[19] S. Schubert and P. Neubert, *What makes visual place recognition easy or hard?* 2021. DOI: 10.48550/ARXIV.2106.12671. [Online]. Available: https://arxiv.org/abs/2106.12671.

[20] J. Engel, T. Schöps and D. Cremers, 'Lsd-slam: Large-scale direct monocular slam,' in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 834–849, ISBN: 978-3-319-10605-2.

[21] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera and J. Berlles, 'S-ptam: Stereo parallel tracking and mapping,' *Robotics and Autonomous Systems*, vol. 93, Apr. 2017. DOI: 10.1016/j.robot.2017.03.019.

[22] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, 'Orb-slam: A versatile and accurate monocular slam system,' *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671.

[23] M. Cummins and P. Newman, 'Appearance-only slam at large scale with fab-map 2.0,' *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011. DOI: 10.1177/0278364910385483. eprint: https://doi.org/10.1177/0278364910385483. [Online]. Available: https://doi.org/10.1177/0278364910385483.

[24] M. Labbé and F. Michaud, 'Memory management for real-time appearance-based loop closure detection,' Sep. 2011, pp. 1271–1276. DOI: 10.1109/IROS.2011.6094602.

[25] W. Maddern, M. Milford and G. Wyeth, 'Cat-slam: Probabilistic localisation and mapping using a continuous appearance-based trajectory,' *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 429–451, 2012. DOI: 10.1177/0278364912438273. [Online]. Available: https://doi.org/10.1177/0278364912438273.

[26] C. Leng, H. Zhang, B. Li, G. Cai, Z. Pei and L. He, 'Local feature descriptor for image matching: A survey,' *IEEE Access*, vol. 7, pp. 6424–6434, 2019. DOI: 10.1109/ACCESS.2018.2888856.

[27] C. Harris and M. Stephens, 'A combined corner and edge detector,' in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[28] J. Shi and Tomasi, 'Good features to track,' in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[29] E. Rosten and T. Drummond, 'Machine learning for high-speed corner detection,' in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443, ISBN: 978-3-540-33833-8.

[30] S. Leutenegger, M. Chli and R. Y. Siegwart, 'Brisk: Binary robust invariant scalable keypoints,' in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[31] N. Dalal and B. Triggs, 'Histograms of oriented gradients for human detection,' in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[32] H. Bay, T. Tuytelaars and L. Van Gool, 'Surf: Speeded up robust features,' in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417, ISBN: 978-3-540-33833-8.

[33] D. G. Lowe, 'Distinctive image features from scale-invariant keypoints,' *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[34] M. Calonder, V. Lepetit, C. Strecha and P. Fua, 'Brief: Binary robust independent elementary features,' in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos and N. Paragios, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792, ISBN: 978-3-642-15561-1.

[35] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, 'Orb: An efficient alternative to sift or surf,' Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[36] Sivic and Zisserman, 'Video google: A text retrieval approach to object matching in videos,' in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, 1470–1477 vol.2. DOI: 10.1109/ICCV.2003.1238663.

[37] A. Angeli, D. Filliat, S. Doncieux and J.-A. Meyer, 'Fast and incremental method for loop-closure detection using bags of visual words,' *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008. DOI: 10.1109/TRO.2008.2004514.

[38] A. Angeli, S. Doncieux, J.-A. Meyer and D. Filliat, 'Incremental vision-based topological slam,' Oct. 2008, pp. 1031–1036. DOI: 10.1109/IROS.2008.4650675.

[39] J. Hartigan, *Clustering Algorithms*. John Wiley and Sons, New York, 1975.

[40] D. Galvez-López and J. D. Tardos, 'Bags of binary words for fast place recognition in image sequences,' *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012. DOI: 10.1109/TRO.2012.2197158.

[41] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988. [Online]. Available: http://portal.acm.org/citation.cfm?id=46712.

[42] L. Di Giammarino, I. Aloise, C. Stachniss and G. Grisetti, *Visual place recognition using lidar intensity information*, 2021. DOI: 10.48550/ARXIV.2103.09605. [Online]. Available: https://arxiv.org/abs/2103.09605.

[43] T. Shan, B. Englot, F. Duarte, C. Ratti and D. Rus, *Robust place recognition using an imaging lidar*, 2021. DOI: 10.48550/ARXIV.2103.02111. [Online]. Available: https://arxiv.org/abs/2103.02111.

[44] Z. Liu, S. Zhou, C. Suo, Y. Liu, P. Yin, H. Wang and Y.-H. Liu, *Lpd-net: 3d point cloud learning for large-scale place recognition and environment analysis*, 2018. DOI: 10.48550/ARXIV.1812.07050. [Online]. Available: https://arxiv.org/abs/1812.07050.

[45] J. Guo, P. V. K. Borges, C. Park and A. Gawel, *Local descriptor for robust place recognition using lidar intensity*, 2018. DOI: 10.48550/ARXIV.1811.12646. [Online]. Available: https://arxiv.org/abs/1811.12646.

[46] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart and C. Cadena, 'Segmatch: Segment based place recognition in 3d point clouds,' in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5266–5272. DOI: 10.1109/ICRA.2017.7989618.

[47] H. Wang, C. Wang and L. Xie, 'Intensity scan context: Coding intensity and geometry relations for loop closure detection,' in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2020. DOI: 10.1109/icra40945.2020.9196764. [Online]. Available: https://doi.org/10.1109%2Ficra40945.2020.9196764.

[48] G. Kim and A. Kim, 'Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map,' in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4802–4809. DOI: 10.1109/IROS.2018.8593953.

[49] W. Zhang and C. Xiao, *Pcan: 3d attention map learning using contextual information for point cloud based retrieval*, 2019. DOI: 10.48550/ARXIV.1904.09793. [Online]. Available: https://arxiv.org/abs/1904.09793.

[50] J. Behley and C. Stachniss, 'Efficient surfel-based slam using 3d laser range data in urban environments,' Jun. 2018. DOI: 10.15607/RSS.2018.XIV.016.

[51] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti and D. Rus, 'Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping,' Oct. 2020, pp. 5135–5142. DOI: 10.1109/IROS45743.2020.9341176.

[52] E. Recherche, E. Automatique, S. Antipolis and Z. Zhang, 'Iterative point matching for registration of free-form curves,' *Int. J. Comput. Vision*, vol. 13, Jul. 1992.

[53] H. Lei, G. Jiang and L. Quan, 'Fast descriptors and correspondence propagation for robust global point cloud registration,' *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3614–3623, 2017. DOI: 10.1109/TIP.2017.2700727.

[54] R. B. Rusu, N. Blodow and M. Beetz, 'Fast point feature histograms (fpfh) for 3d registration,' in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[55] L. Bernreiter, L. Ott, J. Nieto, R. Siegwart and C. Cadena, 'PHASER: A robust and correspondence-free global pointcloud registration,' *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 855–862, Apr. 2021. DOI: 10.1109/lra.2021.3052418. [Online]. Available: https://doi.org/10.1109%2Flra.2021.3052418.

[56] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas and T. Birdal, *Learning multiview 3d point cloud registration*, 2020. DOI: 10.48550/ARXIV.2001.05119. [Online]. Available: https://arxiv.org/abs/2001.05119.

[57] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, 'Vision meets robotics: The kitti dataset,' *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, Sep. 2013. DOI: 10.1177/0278364913491297.

[58] A. S. Huang, M. Antone, E. Olson, L. Fletcher, D. Moore, S. Teller and J. Leonard, 'A high-rate, heterogeneous data set from the darpa urban challenge,' *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1595–1601, 2010. DOI: 10.1177/0278364910384295. eprint: https://doi.org/10.1177/0278364910384295. [Online]. Available: https://doi.org/10.1177/0278364910384295.

[59] C. J. Hegarty and E. Chatre, 'Evolution of the global navigation satellitesystem (gnss),' *Proceedings of the IEEE*, vol. 96, no. 12, pp. 1902–1917, 2008. DOI: 10.1109/JPROC.2008.2006090.

[60] M. Karaim, M. Elsheikh and A. Noureldin, 'Gnss error sources,' in. May 2018, https://www.intechopen.com/books/multifunctional–operation, ISBN: ISBN: 978-1-78923-215-8. DOI: 10.5772/intechopen.75493.

[61] A. Dey and D. Rao, 'Study and analysis of differential gnss and precise point positioning,' *IOSR Journal of Electrical and Electronics Engineering*, vol. 9, pp. 53–59, 2014.

[62] Y. Feng and J. Wang, 'Gps rtk performance characteristics and analysis,' *Journal of Global Positioning Systems*, vol. 7, Jun. 2008. DOI: 10.5081/jgps.7.1.1.

[63] A. Oxley, 'Chapter 5 - gps modernization,' in *Uncertainties in GPS Positioning*, A. Oxley, Ed., Academic Press, 2017, pp. 71–80, ISBN: 978-0-12-809594-2. DOI: https://doi.org/10.1016/B978-0-12-809594-2.00005-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128095942000058.

[64] P. J. Teunissen, 'Carrier phase integer ambiguity resolution,' in *Springer Handbook of Global Navigation Satellite Systems*, P. J. Teunissen and O. Montenbruck, Eds. Cham: Springer International Publishing, 2017, pp. 661–685, ISBN: 978-3-319-42928-1. DOI: 10.1007/978-3-319-42928-1_23. [Online]. Available: https://doi.org/10.1007/978-3-319-42928-1_23.

[65] P. Teunissen and S. Verhagen, 'Gnss ambiguity resolution: When and how to fix or not to fix?' In. Jan. 2008, vol. 132, pp. 143–148, ISBN: 978-3-540-74583-9. DOI: 10.1007/978-3-540-74584-6_22.

[66] C. Rizos, 'Network rtk research and implementation: A geodetic perspective,' *Journal of Global Positioning Systems*, vol. 1, pp. 144–150, Dec. 2002. DOI: 10.5081/jgps.1.2.144.

[67] V. Janssen and J. Haasdyk, 'Assessment of network rtk performance using corsnet-nsw,' Nov. 2011.

[68] R. W. Gerchberg, 'Super-resolution through error energy reduction,' *Journal of Modern Optics*, vol. 21, pp. 709–720, 1974.

[69] G. Huszka and M. A. Gijs, 'Super-resolution optical imaging: A comparison,' *Micro and Nano Engineering*, vol. 2, pp. 7–28, 2019, ISSN: 2590-0072. DOI: https://doi.org/10.1016/j.mne.2018.11.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590007218300157.

[70] G. Cristobal, P. Schelkens and H. Thienpont, *Optical and Digital Image Processing. Fundamentals and Applications*. Apr. 2011. DOI: 10.1002/9783527635245.

[71] H. Greenspan, 'Super-resolution in medical imaging,' *Comput. J.*, vol. 52, pp. 43–63, Jan. 2009. DOI: 10.1093/comjnl/bxm075.

[72] L. Zhang, H. Zhang, H. Shen and P. Li, 'A super-resolution reconstruction algorithm for surveillance images,' *Signal Processing*, vol. 90, no. 3, pp. 848–859, 2010, ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2009.09.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168409003776.

[73] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or and P.-A. Heng, *Pu-net: Point cloud upsampling network*, 2018. DOI: 10.48550/ARXIV.1801.06761. [Online]. Available: https://arxiv.org/abs/1801.06761.

[74] R. Li, X. Li, C.-W. Fu, D. Cohen-Or and P.-A. Heng, 'Pu-gan: A point cloud upsampling adversarial network,' Oct. 2019, pp. 7202–7211. DOI: 10.1109/ICCV.2019.00730.

[75] H. Huang, S. Wu, M. Gong, D. Cohen-Or and U. Ascher, 'Edge-aware point set resampling,' *ACM Transactions on Graphics*, vol. 32, Jan. 2013. DOI: 10.1145/2421636.2421645.

[76] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. Silva, 'Computing and rendering point set surfaces,' *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, 2003. DOI: 10.1109/TVCG.2003.1175093.

[77] Y. Lipman, D. Cohen-Or, D. Levin and H. Tal-Ezer, 'Parameterization-free projection for geometry reconstruction,' *ACM Trans. Graph.*, vol. 26, p. 22, Jul. 2007. DOI: 10.1145/1275808.1276405.

[78] W. Yifan, S. Wu, H. Huang, D. Cohen-Or and O. Sorkine-Hornung, *Patch-based progressive 3d point set upsampling*, 2018. DOI: 10.48550/ARXIV.1811.11286. [Online]. Available: https://arxiv.org/abs/1811.11286.

[79] T. Shan, J. Wang, F. Chen, P. Szenher and B. Englot, *Simulation-based lidar super-resolution for ground vehicles*, 2020. DOI: 10.48550/ARXIV.2004.05242. [Online]. Available: https://arxiv.org/abs/2004.05242.

[80] H. Chen, X. He, L. Qing, Y. Wu, C. Ren and C. Zhu, *Real-world single image super-resolution: A brief review*, 2021. arXiv: 2103.02368 [eess.IV].

[81] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue and Q. Liao, 'Deep learning for single image super-resolution: A brief review,' *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3106–3121, Dec. 2019, ISSN: 1941-0077. DOI: 10.1109/tmm.2019.2919431. [Online]. Available: http://dx.doi.org/10.1109/TMM.2019.2919431.

[82] C. E. Shannon, 'Communication in the presence of noise,' *Proceedings of the IEEE*, vol. 86, pp. 447–457, 1998.

[83] S. Gohshi, 'The relation between super resolution and aliasing and how to overcome its limitations,' in *2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, 2015, pp. 25–29. DOI: 10.1109/ISPACS.2015.7432730.

[84] K. Zhang, J. Liang, L. V. Gool and R. Timofte, *Designing a practical degradation model for deep blind image super-resolution*, 2021. arXiv: 2103.14006 [eess.IV].

[85] C. Dong, C. C. Loy, K. He and X. Tang, *Image super-resolution using deep convolutional networks*, 2015. DOI: 10.48550/ARXIV.1501.00092. [Online]. Available: https://arxiv.org/abs/1501.00092.

[86] A. Horé and D. Ziou, 'Image quality metrics: Psnr vs. ssim,' Aug. 2010, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[87] Z. Wang, A. Bovik, H. Sheikh and E. Simoncelli, 'Image quality assessment: From error visibility to structural similarity,' *Image Processing, IEEE Transactions on*, vol. 13, pp. 600–612, May 2004. DOI: 10.1109/TIP.2003.819861.

[88] Y. Tai, J. Yang and X. Liu, 'Image super-resolution via deep recursive residual network,' in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2790–2798. DOI: 10.1109/CVPR.2017.298.

[89] J. Kim, J. K. Lee and K. M. Lee, *Accurate image super-resolution using very deep convolutional networks*, 2015. DOI: 10.48550/ARXIV.1511.04587. [Online]. Available: https://arxiv.org/abs/1511.04587.

[90] K. He and J. Sun, *Convolutional neural networks at constrained time cost*, 2014. DOI: 10.48550/ARXIV.1412.1710. [Online]. Available: https://arxiv.org/abs/1412.1710.

[91] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert and Z. Wang, *Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network*, 2016. DOI: 10.48550/ARXIV.1609.05158. [Online]. Available: https://arxiv.org/abs/1609.05158.

[92] B. Lim, S. Son, H. Kim, S. Nah and K. M. Lee, *Enhanced deep residual networks for single image super-resolution*, 2017. DOI: 10.48550/ARXIV.1707.02921. [Online]. Available: https://arxiv.org/abs/1707.02921.

[93] S. Albawi, T. A. Mohammed and S. Al-Zawi, 'Understanding of a convolutional neural network,' in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[94] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, 2016. DOI: 10.48550/ARXIV.1603.07285. [Online]. Available: https://arxiv.org/abs/1603.07285.

[95] C. Dong, C. C. Loy and X. Tang, *Accelerating the super-resolution convolutional neural network*, 2016. DOI: 10.48550/ARXIV.1608.00367. [Online]. Available: https://arxiv.org/abs/1608.00367.

[96] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016, ISBN: 9780262035613.

[97] A. F. Agarap, *Deep learning using rectified linear units (relu)*, 2018. DOI: 10.48550/ARXIV.1803.08375. [Online]. Available: https://arxiv.org/abs/1803.08375.

[98] R. Pascanu, T. Mikolov and Y. Bengio, *On the difficulty of training recurrent neural networks*, 2012. DOI: 10.48550/ARXIV.1211.5063. [Online]. Available: https://arxiv.org/abs/1211.5063.

[99] L. Lu, 'Dying ReLU and initialization: Theory and numerical examples,' *Communications in Computational Physics*, vol. 28, no. 5, pp. 1671–1706, Jun. 2020. DOI: 10.4208/cicp.oa-2020-0165. [Online]. Available: https://doi.org/10.4208%2Fcicp.oa-2020-0165.

[100] A. L. Maas, 'Rectifier nonlinearities improve neural network acoustic models,' 2013.

[101] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio and P. Vincent, 'The difficulty of training deep architectures and the effect of unsupervised pretraining.,' *Journal of Machine Learning Research - Proceedings Track*, vol. 5, pp. 153–160, Jan. 2009.

[102] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980.

[103] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting,' *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jun. 2014.

[104] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. DOI: 10.48550/ARXIV.1502.03167. [Online]. Available: https://arxiv.org/abs/1502.03167.

[105] (2017). 'DBoW3,' [Online]. Available: https://github.com/rmsalinas/DBow3 (visited on 25/04/2022).

[106] G. Bradski, 'The OpenCV Library,' *Dr. Dobb's Journal of Software Tools*, 2000.

[107] T. Shan, *Lidar super-resolution*, https://github.com/RobustFieldAutonomyLab/lidar_super_resolution. (visited on 20/04/2022).

[108] O. Ronneberger, P. Fischer and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. DOI: 10.48550/ARXIV.1505.04597. [Online]. Available: https://arxiv.org/abs/1505.04597.

[109] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, 2015. DOI: 10.48550/ARXIV.1506.02142. [Online]. Available: https://arxiv.org/abs/1506.02142.

[110] X. Yue, B. Wu, S. A. Seshia, K. Keutzer and A. L. Sangiovanni-Vincentelli, *A lidar point cloud generator: From a virtual world to autonomous driving*, 2018. DOI: 10.48550/ARXIV.1804.00103. [Online]. Available: https://arxiv.org/abs/1804.00103.

[111] Ouster, *Software user manual*, https://data.ouster.io/downloads/software-user-manual/software-user-manual-v2p0.pdf, 2021. (visited on 04/06/2022).

[112] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, *Photo-realistic single image super-resolution using a generative adversarial network*, 2016. DOI: 10.48550/ARXIV.1609.04802. [Online]. Available: https://arxiv.org/abs/1609.04802.

[113] L. Zhang, M. Camurri and M. Fallon, *Multi-camera lidar inertial extension to the newer college dataset*, 2021. arXiv: 2112.08854 [cs.RO].

[114] G. Weber, D. Dettmering and H. Gebhard, 'Networked transport of rtcm via internet protocol (ntrip),' in. Jan. 2005, pp. 60–64, ISBN: 3-540-24055-1. DOI: 10.1007/3-540-27432-4_11.

[115] O. Robotics, *Ros - robot operating system*, https://www.ros.org/. (visited on 12/05/2022).

[116] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, 'Tensorflow: A system for large-scale machine learning,' in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[117] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, J. J. Yebes and S. Gámez, 'Bidirectional loop closure detection on panoramas for visual navigation,' in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 1378–1383. DOI: 10.1109/IVS.2014.6856457.

[118] Ouster, *Compare our gen 2 sensors*, https://ouster.com/products/. (visited on 14/05/2022).

[119] N. Chervyakov, P. Lyakhov and N. Nagornov, 'Analysis of the quantization noise in discrete wavelet transform filters for 3d medical imaging,' *Applied Sciences*, vol. 10, no. 4, 2020, ISSN: 2076-3417. [Online]. Available: https://www.mdpi.com/2076-3417/10/4/1223.

[120] M. S. M. Sajjadi, B. Schölkopf and M. Hirsch, *Enhancenet: Single image super-resolution through automated texture synthesis*, 2016. DOI: 10.48550/ARXIV.1612.07919. [Online]. Available: https://arxiv.org/abs/1612.07919.

[121] T. Köhler, M. Bätz, F. Naderi, A. Kaup, A. Maier and C. Riess, *Toward bridging the simulated-to-real gap: Benchmarking super-resolution on real data*, 2018. DOI: 10.48550/ARXIV.1809.06420. [Online]. Available: https://arxiv.org/abs/1809.06420.

[122] P. Foster, Z. Sun, J. J. Park and B. Kuipers, 'Visagge: Visible angle grid for glass environments,' in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2213–2220. DOI: 10.1109/ICRA.2013.6630875.

Appendix A

Additional Material

A.1 Lidar Super Resolution Architecture

The full lidar super resolution network architecture by Shan et. al from [79] is shown in Table A.3, where the "Convolutional block" and "Up-Block" used in the network are given in Table A.1 and Table A.2 respectively.

Layer	Type	Parameters
1	Convolutional layer	X filters, 3x3 kernel size, "same"-padding
2	Batch Normalization	
3	ReLu	
4	Convolutional layer	X filters, 3x3 kernel size, "same"-padding
5	Batch Normalization	
6	ReLu	

Table A.1: Overview of the "convolution block" used in the lidar super resolution network by Shan et al. [79], where X is an input parameter.

Layer	Type	Parameters
1	Transpose Convolutional layer	X filters, "same"-padding, (Y,Z) stride
2	Batch Normalization	
3	ReLu	

Table A.2: Overview of the "Up-Block" used in the lidar super resolution network by Shan et al. [79], where X is an input parameters.

Layer	Type	Parameters
0*	Up-Block	64 filters, 2x1 stride.
1	Convolutional Block	64 filters
2	Average Pooling	2x2 stride
3	Dropout	25% dropout rate
4	Convolutional Block	128 filters
5	Average Pooling	2x2 stride
6	Dropout	25% dropout rate
7	Convolutional Block	256 filters
8	Average Pooling	2x2 stride
9	Dropout	25% dropout rate
10	Convolutional Block	512 filters
11	Average Pooling	2x2 stride
12	Dropout	25% dropout rate
13	Convolutional Block	1024 filters
14	Dropout	25%
15	Up-block	512 filters, 2x2 stride
16	Concatenate	Layer 10 output and layer 15 output
17	Convolutional Block	512 filters
18	Dropout	25% dropout rate
19	Up-block	256 filters, 2x2 stride
20	Concatenate	Layer 7 output and layer 19 output
21	Convolutional Block	256 filters
22	Dropout	25% dropout rate
23	Up-block	128 filters, 2x2 stride
24	Concatenate	Layer 4 output and layer 23 output
25	Convolutional Block	128 filters
26	Dropout	25% dropout rate
27	Up-block	64 filters, 2x2 stride
28	Concatenate	Layer 1 output and layer 27 output
29	Convolutional Block	64 filters
30	Convolutional Layer	1 filter, 1x1 kernel size, ReLU activation function

Table A.3: Overview of the lidar super resolution network architecture by Shan et al. [79]. The definition of the convolutional block and the up-block are given in Tables A.1 and A.2 respectively. *Note that the upsampling in layer 0 is repeated N times based on the desired upscaling factor as in Equation (3.1).

Grunde Nyvoll

Exploiting Lidar Imaging for Robot Perception and Place Recognition

NTNU
Norwegian University of
Science and Technology