Influence of Complex Environments on LiDAR-Based Robot Navigation

Mémoire

Sébastien Michaud

Maîtrise en informatique
Maître ès sciences (M.Sc.)

Québec, Canada

Résumé

La navigation sécuritaire et efficace des robots mobiles repose grandement sur l'utilisation des capteurs embarqués. L'un des capteurs qui est de plus en plus utilisé pour cette tâche est le *Light Detection And Ranging* (LiDAR). Bien que les recherches récentes montrent une amélioration des performances de navigation basée sur les LiDARs, faire face à des environnements non structurés complexes ou des conditions météorologiques difficiles reste problématique. Dans ce mémoire, nous présentons une analyse de l'influence de telles conditions sur la navigation basée sur les LiDARs. Notre première contribution est d'évaluer comment les LiDARs sont affectés par les flocons de neige durant les tempêtes de neige. Pour ce faire, nous créons un nouvel ensemble de données en faisant l'acquisition de données durant six précipitations de neige. Une analyse statistique de ces ensembles de données, nous caractérisons la sensibilité de chaque capteur et montrons que les mesures de capteurs peuvent être modélisées de manière probabilistique. Nous montrons aussi que les précipitations de neige ont peu d'influence au-delà de 10 m. Notre seconde contribution est d'évaluer l'impact de structures tridimensionnelles complexes présentes en forêt sur les performances d'un algorithme de reconnaissance d'endroits. Nous avons acquis des données dans un environnement extérieur structuré et en forêt, ce qui permet d'évaluer l'influence de ces derniers sur les performances de reconnaissance d'endroits. Notre hypothèse est que, plus deux balayages laser sont proches l'un de l'autre, plus la croyance que ceux-ci proviennent du même endroit sera élevée, mais modulé par le niveau de complexité de l'environnement. Nos expériences confirment que la forêt, avec ses réseaux de branches compliqués et son feuillage, produit plus de données aberrantes et induit une chute plus rapide des performances de reconnaissance en fonction de la distance. Notre conclusion finale est que, les environnements complexes étudiés influencent négativement les performances de navigation basée sur les LiDARs, ce qui devrait être considéré pour développer des algorithmes de navigation robustes.

Abstract

To ensure safe and efficient navigation, mobile robots heavily rely on their ability to use on-board sensors. One such sensor, increasingly used for robot navigation, is the *Light Detection And Ranging* (LiDAR). Although recent research showed improvement in LiDAR-based navigation, dealing with complex unstructured environments or difficult weather conditions remains problematic. In this thesis, we present an analysis of the influence of such challenging conditions on LiDAR-based navigation. Our first contribution is to evaluate how LiDARs are affected by snowflakes during snowstorms. To this end, we create a novel dataset by acquiring data during six snowfalls using four sensors simultaneously. Based on statistical analysis of this dataset, we characterized the sensitivity of each device and showed that sensor measurements can be modelled in a probabilistic manner. We also showed that falling snow has little impact beyond a range of 10 m. Our second contribution is to evaluate the impact of complex of three-dimensional structures, present in forests, on the performance of a LiDAR-based place recognition algorithm. We acquired data in structured outdoor environment and in forest, which allowed evaluating the impact of the environment on the place recognition performance. Our hypothesis was that the closer two scans are acquired from each other, the higher the belief that the scans originate from the same place will be, but modulated by the level of complexity of the environments. Our experiments confirmed that forests, with their intricate network of branches and foliage, produce more outliers and induce recognition performance to decrease more quickly with distance when compared with structured outdoor environment. Our conclusion is that falling snow conditions and forest environments negatively impact LiDAR-based navigation performance, which should be considered to develop robust navigation algorithms.

Contents

List of Tables

List of Figures

Acronyms

2D

Two Dimensional. 2, 5, 7, 12, 17, 30, 46

3D

Three Dimensional. vii, 2, 7, 8, 12, 13, 15, 17, 30, 32, 35, 36, 40–42, 44

BoW

Bag of Words. vii, 8, 40–42, 44, 53

FN

False Negative. 49

FOV

Field Of View. 32, 33, 52

FP

False Positive. 49–53, 57

GPS

Global Positioning System. 1, 28, 30

GUI

Graphical User Interface. 28

ICP

Iterative Closest Point. ix, 46–48, 53

IMU

Inertial Measurements Units. 1, 28, 30

LiDAR

Light Detection And Ranging. vii, ix, 2, 3, 5, 6, 8, 11–17, 19, 23, 26–30, 33, 36, 41, 44, 49, 51–53, 55–57

NDT

Normal Distribution Transform. 8

PTU

Pan-Tilt Unit. 12, 15, 28–30, 52

RGB

Red, Green, Blue colour. 2, 17, 18

RGB-D

Red, Green, Blue plus Depth. 7

ROS

Robot Operating System. 17, 30

SLAM

Simultaneous Localization And Mapping. 5, 8, 27, 49, 52, 56

SSH

Secure Shell. 28

TN

True Negative. 49

TP

True Positive. 49

UDP

User Datagram Protocol. 32

UGV

Unmanned Ground Vehicle. ix, 5, 28, 32

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Prof. Philippe Giguère and Prof. Jean-François Lalonde, for the continuous support during my Master studies. They seamlessly provided funding and research equipment, they assisted me during the writing, but most of all, they shared their precious knowledge and time to help me reach my goals.

A very special thanks goes out to François Pomerleau, a Postdoctoral Fellow who taught me a lot about the world of research and helped me learn the essential tools for applied robotics. He always answered my numerous questions patiently and provided me with useful tips for my work, even after leaving our research lab.

I would like to thank my fellow lab mates for the help they gave me on multiple projects, the stimulating discussions we had and the fun time we shared during social activities.

Introduction

Modern robotics has experienced tremendous growth since its inception during the Industrial Revolution. Whether it is to assist humans in their work, to automate some tasks or to perform dangerous tasks, robots are emerging in a wide variety of applications. Recent technology such as the self-driving car project from Google and the BigDog quadruped robot from Boston Dynamics are feats of engineering that show how robots have evolved from being able to operate in controlled environments to performing complex tasks in a challenging or unpredictable situations.

One of the key elements that allowed for such progress is the ability of the robot to adapt to changing environments. This is especially true for mobile robotics, where not only the surrounding environment can change, but the robot itself can move and must therefore be able to locate itself. This capability highly depends on algorithms that convert the raw sensor inputs into a convenient representation or abstraction of the environment. This concept is known as artificial perception.

A wide range of sensors are available to assist robots navigation. It is possible, for instance, to estimate the relative movements of the robot using wheel encoders. Unfortunately, when the ground friction coefficient is variable or the load-bearing surface is very uneven, pose estimation relying solely on this sensor is highly unreliable. Similarly, *Inertial Measurements Units (IMU)* are composed of accelerometers and gyroscopes that can be used to infer changes in position. Although these sensors can be very precise over short distances, they accumulate errors over time, which inevitably leads to a drift on the pose estimation.

The natural solution to this problem is to use sensors capable of providing absolute positioning. Arguably the most popular sensor providing such information is the *Global Positioning System (GPS)*. However, there are inherent problems with this sensor as well. *GPS* requires receiving the signal of at least three satellites at all time. These signals can be blocked by building, terrain, dense foliage or other structures, thus causing important positioning errors or possibly no positioning at all.

Alternatively, a global map of landmarks can be used by the robot to locate itself. These landmarks are distinctive features acquired using sensors that react to external stimuli of the robot environment (i.e. exteroceptive sensors). Visual markers acquired with cameras, sounds signatures obtained with microphones or singular structures detected with sonars are examples of such features. In addition to solving localization and mapping problems, the use of such features make it possible to perform many

other navigation tasks. For instance, it is possible to classify ground type to predict traversability or detect obstacles to compute path planning.

The most widely used sensors for such task is the conventional *Red, Green, Blue colour (RGB)* camera. In addition to its generally low price, cameras provide valuable appearance information about the scene in which the robot operates. This information can be processed, for instance, to analyze geometric elements of the scene or identify objects in it. Despite these obvious advantages, the images obtained by the cameras are produced by a projection and it is difficult or impossible to retrieve information regarding the three-dimensional structure of the scene. Furthermore, the quality of acquisitions depends greatly on the lighting conditions of the environment. The processing of these images for real-time navigation requires powerful hardware, which is not always available on the robot.

The *Light Detection And Ranging (LiDAR)* is another sensor that provides valuable data about the scene. It uses a laser to measure distances at different angles from the sensor centre. Most *LiDAR*s provide *Two Dimensional (2D)* set of points, but some are able to directly produce *Three Dimensional (3D)* data, called point clouds. *LiDAR*s are generally more expensive than cameras, but the geometrical information obtained therewith is often complementary to the appearance information provided by the cameras. For instance, a *LiDAR* will faithfully report the flat structure of a white wall, while the lack of visual features will make it impossible for the camera to infer such information. On the other hand, a camera would be able to locate itself using the rich appearance information of a poster on a wall, but the geometrical information of the wall retrieved by the *LiDAR* is of little help for the localization task. Another important difference between those sensors is that the *LiDAR* is an active sensor that can be used day and night and which is mostly unaffected by lighting conditions, while the camera is a passive sensor for which the compensation for changes in lighting conditions is among the most challenging problems.

In Chapter 1 we will review existing literature about robot navigation. Because cameras were used in the field before *LiDAR*s and because those sensors are somehow similar, techniques used with cameras inspired those for *LiDAR*s. For this reason, there will be references related to cameras, but *LiDAR*s proved to be an excellent choice for robot navigation and will be the sensors of interest of this document. As we will see, existing works for *LiDAR* mainly deal with simpler situations such as structured indoor or semi-structured city environments. We will therefore focus our attention on more challenging environments such as falling snow conditions or highly unstructured outdoor environments. More precisely, we are interested in the impact of those complex environments on the resulting data and the task to be achieved.

Subsequently, in Chapter 2, we will briefly present the basics of how *LiDAR*s operate and how readings are theoretically affected when scanning small structures or dynamic objects. These conditions are likely to be found in many complex environments where the robot might need to navigate. Therefore, we are interested in quantifying the impact such small structures and dynamic objects on the sensor readings. For that matter, we placed four different *LiDAR*s so as to acquire data during falling snow

conditions. Using our experimental setup, we are able to evaluate the influence of snowflakes of various sizes and falling at different rates on the sensors measurements.

Chapter 3 provides a higher level analysis of the influence of the environment on a navigation algorithm. In this case, we chose a state-of-the-art *LiDAR*-based place recognition algorithm and we compared the results obtained in different environments. We created our own dataset in conditions similar to the one presented in the original article and acquired another dataset in forested area. Because forests are composed of multiple small structures such as branches and leaves, we considered the latter dataset more challenging, and found that the radius within which it is possible to recognize places reliably is lower in such environments.

Chapter 1

Literature Review

1.1 Introduction

Mobile robotics literature contains a wide variety of *LiDAR*-based solutions to navigation problems. This includes using *2D* for indoor *Simultaneous Localization And Mapping (SLAM)* [Grisetti et al., 2007, Kohlbrecher et al., 2011], geolocation in forest [Hussein et al., 2015] or detection of traversable grass-like vegetation using laser remission [Wurm et al., 2009]. As it is the case for this thesis, some studies focus their efforts on finding solutions for challenging conditions (or environments). It should be noted that there is no clear definition of what constitutes *challenging* conditions, as it varies between sensors. A good understanding of the *LiDAR* functioning [Amann et al., 2001] therefore helps understand how it applies to this sensor (see Section 2.2). The Marulan Data Sets [Peynot et al., 2010] contains good examples of such conditions, including natural outdoor environments and area with presence of smoke, dust and rain.

In this thesis, we are interested in the influence of such difficult situations, either due to complex unstructured environments or challenging weather conditions, on *LiDAR*-based robot navigation. In Section 1.2, we will discuss research related to navigation in snowfall conditions, which is the linked to Chapter 2. Subsequently, Section 1.3 will present papers related to feature-based place recognition and navigation in forest environments, which are more closely related to Chapter 3.

1.2 Snowfall Conditions

Snowfall conditions are challenging as snowflakes cause occlusion or interference with the laser beam. Because of their small size and dynamic nature, they tend to produce a signal similar to random noise in sensors acquisitions. While it is often possible to avoid navigating in these conditions, some robots will inevitably face this situation in order to perform the tasks for which they were designed.

For instance, Moorehead et al. [1999] aimed at developing a robot to search and classify meteorites in Antartica. In their experiments, the *Unmanned Ground Vehicle (UGV)* droves autonomously for

10.3 km in different weather and terrain conditions. The navigation was based on stereo camera images and single line *LiDAR* scans. In this article, the authors explain that the snow-covered surfaces did not provide a lot of visual cues, which made stereo cameras unreliable. For this reason, the robot took advantage of the *LiDAR*, mainly for obstacle avoidance, but often as single sensor. In return, the authors stated that part of the experiments "was performed during heavy snow which made the laser useless", and therefore used alternative solutions. As we will see in Chapter 2, new *LiDAR* technologies help reduce the impact of small particles on the readings.

Another example of robots which need to handle all-weather conditions are those deployed on the battlefield. Yamauchi [2010] uses ultra-wideband radar, stereo camera and *LiDAR* data for navigation. According to the author, "[*LiDAR*] and stereo vision provide greater accuracy and resolution in clear weather but has difficulty with precipitation and obscurants". The ultra-wideband radar has the ability to see through small particles such as snow, rain and fog and is therefore complementary to other sensors used. The final system achieve good results by using traditional sensor fusion, as well as a selective use of the sensors, which are activated or deactivated depending on the conditions.

Sumi et al. [2013], for their part, aimed at evaluating sensors for personal care robots in natural lighting and falling snow conditions. For that matter, they built two simulators that reproduced those conditions and used three types of sensor: an active stereo sensor (Microsoft Kinect), a *LiDAR* (Mesa SR4000) and a passive stereo sensor (PGR Bumblebee2). This approach is interesting since it allowed to control various parameters that may affect the sensor readings. However, the physical properties of simulated snowflakes can be different from that of real snowflakes, leading to inaccurate results for real applications. By contrast, as we used natural events to estimate the impact of snow flakes on *LiDAR* measurements, we were not able to control the environment parameters.

The work by Barnum et al. [2010] is probably the most similar to our work presented in Chapter 2, but applied to video rather than *LiDAR*s data. They explain that rain and snow are dynamic process, which causes spatial and temporal fluctuations in videos. Although the spatio-temporal changes appear to be chaotic, they were able to predict the overall effect of these conditions on the video, in frequency space. This modelling of the effect of weather conditions on videos improved the noise filtering for features extraction, when compared to previous pixel-based or patch-based methods.

Finally, Servomaa et al. [2002] have installed a set of sensors for snowfall observation. The system was composed of a radar and a *LiDAR* that recorded the atmospheric profile up to 6000 m and another radar along with two balances to record snowfall at ground level. Although the installation was fixed and the *LiDAR* was used only to evaluate transition in cloud conditions, they proposed techniques to estimate snowfall characteristics. Although not specifically mentioned in their paper, these techniques may advantageously be used to adapt the behaviour of the robot depending on weather conditions.

1.3 Feature-Based Place Recognition and Forest Environments

As we will see in more details in Chapter 3, place recognition is a useful tool for mobile robot navigation. A robot can determine whether he is in previously visited place or not by comparing its current sensor acquisition with those acquired earlier. Since the relevant information density in the raw sensor data is low, data is usually converted into a representation that better capture the key information. This process is known as features extraction and consist in identifying points of interest from the sensor data (e.g. image), called feature keypoints, and create a descriptor for each keypoint. A descriptor is a vector containing values which should represent the area surrounding the keypoint robustly, even under small disturbance such as different lighting conditions or change of viewpoint.

The popularity of using features representation can be attributed to the field of computer vision, especially with the introduction of SIFT [Lowe, 2004] and SURF [Bay et al., 2006]. This concept has since been applied to *3D* data. Table 1.1 present some popular *2D* and *3D* features.

Although the choice of features can influence the performance of place recognition, we will not address this issue directly in this thesis. Instead, we analyze the impact of some environments on the overall place recognition performance. For those further interested in this topic, a number of articles that present comparative evaluation of different features are available in the literature. For instance, Filipe and Alexandre [2014] present an evaluation of *3D* keypoint detectors, more precisely for *Red, Green, Blue plus Depth (RGB-D)* objects. In this paper, they focus on the invariance of keypoints detectors to rotations, scales and translation. Similarly, Boyer et al. [2011] propose a benchmark to estimate how different algorithms perform for retrieving keypoints and descriptors when subject to different geometric transformations.

Type of data	**Keypoint/descriptor**	**Name**	**Reference**
Image	Keypoint	Harris and Stephens corners	[Harris and Stephens, 1988]
Image	Both	SIFT	[Lowe, 2004]
Image	Both	SURF	[Bay et al., 2006]
3D	Descriptor	FPFH	[Rusu et al., 2009]
3D	Both	ISS	[Yu, 2009]
3D	Descriptor	SHOT	[Tombari et al., 2010]
3D	Both	NARF	[Steder et al., 2011a]

Table 1.1 – Examples of popular descriptors and keypoints detectors for images and *3D* data. Some *2D* keypoints have been adapted for *3D* such as Harris and Stephens and SIFT.

Several techniques have been proposed to solve the place recognition problem, but most approaches use cameras as primary sensor [Torralba et al., 2003, Ulrich and Nourbakhsh, 2000]. The most noteworthy example is probably the work of Cummins and Newman [2008], commonly referred as FAB-MAP. They used a probabilistic framework to recognize previously seen places and identify new

places. The algorithm was able to recognize the redundant visual information that did not significantly help to distinguish places, which was used to reduce the probability of such examples to be labeled as originating from the same place. They reach recall of 48 % at 100 % precision for a dataset of 1.9 km. The authors also proposed an enhanced version of the algorithm [Cummins and Newman, 2011] in which they focus on scaleability primarily using the concept of inverted index. They are able to reach 48 % recall at 100 % precision on a 70 km dataset. As we will see in Chapter 3, place recognition is often used to detect loop closures for *SLAM*. In this scenario, the presence of false positives is often catastrophic, which is why they present the results for a precision of 100 %.

By contrast, existing place recognition algorithms based on *3D LiDAR* data only are quite limited. To the best of our knowledge, Magnusson et al. [2009] are the first to address this problem. They used *Normal Distribution Transform (NDT)* to create feature histograms based on surface orientation and smoothness to represent each scan. They also aligned scans with respect to dominant surface orientation to achieve rotation invariance. Finally, they used expectation maximization to automatically determine the threshold that separated corresponding from non-corresponding scans. They achieved recall rates between 22.9 % and 69.6 % with false positive rates below 1.17 %, for three different datasets. More recently, Röhling et al. [2015] proposed a similar approach for solving the place recognition problem based on *3D* lidar data. The authors indicated that they used simpler histograms and a different distance metric to achieve similar results to Magnusson et al. [2009]. The main contribution of their algorithm is that it is simpler to implement and less computationally demanding.

The two previously mentioned algorithms used global descriptors (i.e. a single descriptor for each scan), which is often faster to process but less robust to local disturbances than approaches based on local features (i.e. a set of feature keypoints and associate descriptors for each scan). For our experiments in Chapter 3, we will use the algorithm proposed by Steder et al. [2011b], which is itself an extension of their previous work in [Steder et al., 2010]. The algorithm, that will be presented in more details in Section 3.3, use a mixture of *Bag of Words (BoW)* and features matching to recognize places.

To our knowledge, none of the previously discussed algorithms have been tested in forest environments or have been developed for specific conditions, but were rather proposed as generic solutions. When the environment in which the robot will operate is known in advance, algorithms can be developed or fine-tuned using this prior knowledge to potentially improve performance. For instance, one could intuitively assume that in forest, tree trunks are more reliable features than foliage. Latulippe et al. [2013] proposed to use machine learning to automatically identify and filter local point cloud features in natural environments to be robust for scans alignment purpose. In their paper, they indeed concluded that features produced in foliage regions are not reliable. Other examples of prior knowledge used for *LiDAR*-based navigation in forest include [Lalonde et al., 2006], which present a technique for segmenting data in three classes and [Mcdaniel et al., 2012], which present a technique for segmenting ground and trees in forest. These segmented regions of data can be used to identify navigable area or be used as features for multiple tasks.

An example algorithm using this type of feature, specific to the forest, is presented in Song et al. [2012a]. They proposed a localization solution using the largest group of approximately parallel tree trunks as features to align successive scans along five dimensions (ignoring the translation relative to the gravity vector). Similarly, Miettinen et al. [2007] created a global map of trees in the form of a graph. In this graph, nodes represented tree trunks and edges represented the distance between those trunks. This representation was then used for localization and mapping, by using best matched sub graphs.

Chapter 2

Snowfall Influence on LiDAR Data

2.1 Introduction

An internal representation of the environment is essential for robots to perform the various tasks for which they are designed. If such representation is not provided beforehand, which is often the case, it will be created using sensors available on the robot. Unfortunately, each sensor acquires a specific type of data in a limited measurement interval. In addition, either because of the sensor itself or because of the acquisition environment, the data obtained are always noisy. While this can be of little influence for some simple problems, ignoring these problems can cause serious misunderstanding of the scene and lead to the failure of the tasks, potentially causing damage to the robot or injuring humans. As we will see in this chapter, *LiDAR*s enable us to assess the three-dimensional structure the environment, but they are especially noisy when measuring dynamic objects, small structures or object edges. Forested area and falling snow conditions are good examples of such challenging environments for *LiDAR*s. Characterizing how *LiDAR*s will react in those conditions will allow us to develop more robust and versatile algorithms.

In this chapter, we will first introduce the basics of *LiDAR*s operations (in section 2.2) to better understand why they are affected by small structures. We will follow up with our main contribution, that is to provide a characterization of the behaviour of four well-known *LiDAR*s in snowy conditions. Through an extensive empirical study performed on a novel dataset captured under varying degrees of snowfall, we evaluate how much these *LiDAR*s are sensitive—or not—to falling snow. We show that recent advances in sensor designs have increased their robustness even to significant snowfall. Section 2.3 describe how data acquisition was performed, section 2.4 present a temporal analysis of the data and section 2.5 describe the distribution of snowflake echoes as a function of range before we conclude in section 2.6.

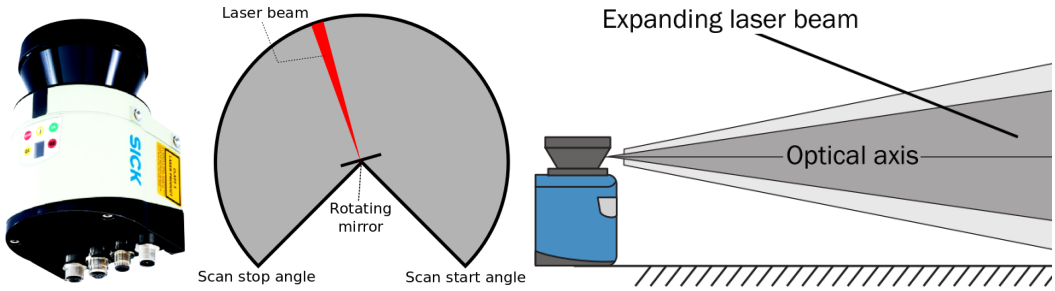


Figure 2.1 – Example of a *LiDAR* device and a simplified representation of the laser trajectory. (a) The SICK LMS151 *LiDAR*. The truncated cone of the upper part is the place from where the laser is projected and is made of a material allowing the laser to pass through unaffected. (b) A simplified representation of how the internal rotating mirror change the scanning angle of the laser. (c) A side view of the laser beam going out of the *LiDAR* (figure modified from [SICK, a]).

2.2 Basics of LiDARs

LiDAR is a technology based on laser time of flight to measure distances. Amann et al. [2001] present the physical details as well as the pros and cons of three time of flight techniques commonly used in *LiDAR*s, namely the pulsed, phase-shift and frequency modulated continuous wave. Although the *LiDAR*s we will use for our research are all based on time of flight, the underlying technique is not specified by the manufacturers. In the context of our research, we focus more on higher level concepts that could cause sensor readings to be erroneous for modelling environments or objects.

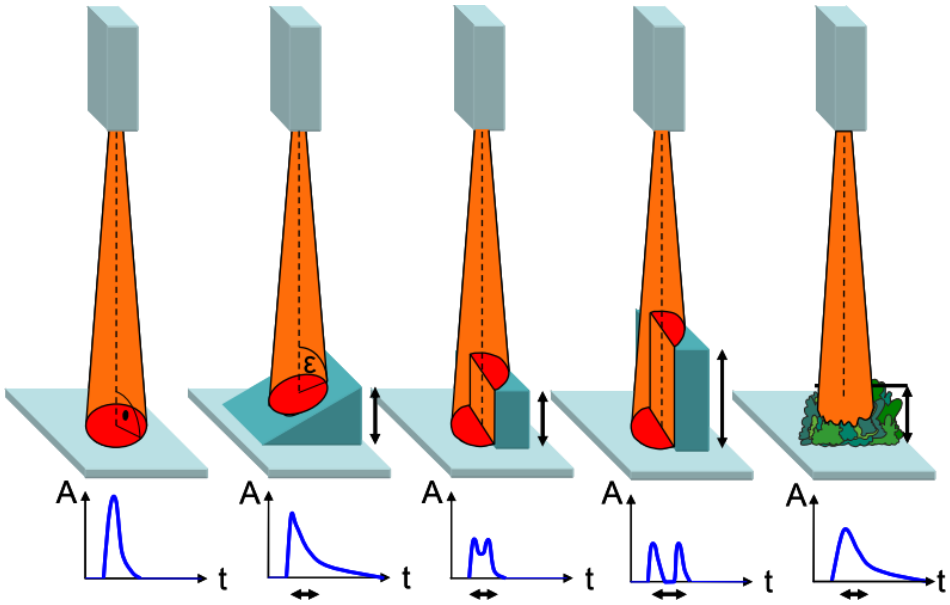
*LiDAR*s generally build *2D* data points using an internal rotating mirror (see Figure 2.1b). It is possible to create a *3D* point cloud by moving the sensor with an external tool (e.g. a *Pan-Tilt Unit (PTU)* or the robot itself) and merging scans. There are some sensors, such as the Velodyne HDL-32E, that directly provide this *3D* information. Given that the points are acquired sequentially, dynamic objects may be distorted in the final representation. Figure 2.3b depicts a point cloud created using the *2D* SICK LMS151 mounted on a *PTU*.

Beams emitted by a *LiDAR* have a given width and angle at source. This causes the beam two-dimensional pattern on the target to grow with distance. Once the light hits the target, it bounces back to the sensor which will extract the range information from it. Obviously, when the target material is highly absorbent or reflective, the light might not reach the sensor, therefore causing missing data points. Otherwise, the sensor will receive the signal which may be represented by a curve of light intensity as function of time. Smooth lambertian surfaces will produce a unimodal distribution from which it is easy to calculate the target range, but multimodal waveforms caused by partially transparent material, fog, dust, small objects and edges lead to an ambiguous interpretation. Figure 2.2 depicts laser beams hitting different targets along with the resulting waveforms. While some *LiDAR*s provide full waveform, they generally only output a single or few echoes and the inference method differs between sensors (e.g., using the first or last waveform peak, using the mean). For this reason, it is

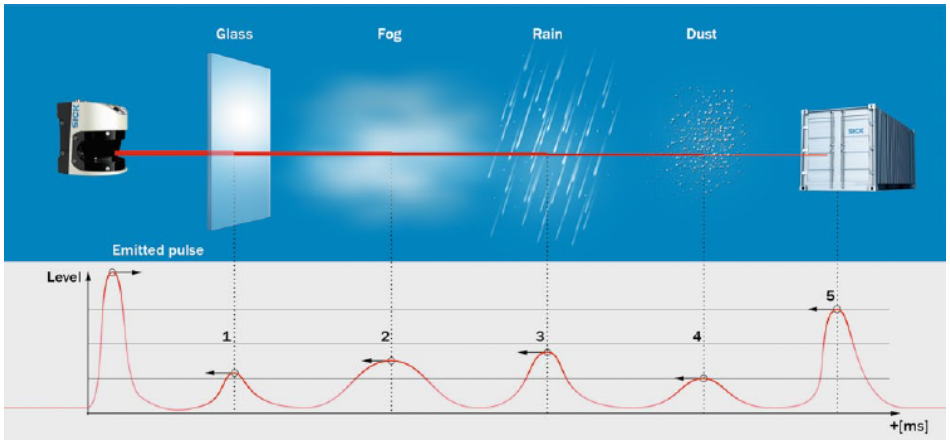
important to determine whether the sensor is well suited to the needs.

In order to visualize more easily the data obtained with a *LiDAR*, they are typically represented by a point cloud. Building such point cloud only consists in converting each distance inferred from a laser return into a point in the *3D* space. Figure 2.3 shows an example of a scene (a) and the corresponding point cloud representation (b). There are also highlighted regions of the point cloud where examples of reading errors caused by the environment surrounding the robot can be seen.

In this document, we focus our attention on the impact of small structures such as the branch presented in Figure 2.3b region **D**. More specifically, the present chapter deals with the impact of falling snow on the raw data of *LiDAR*s. In the next section (Section 2.3), we will explain how we gathered data for this analysis.



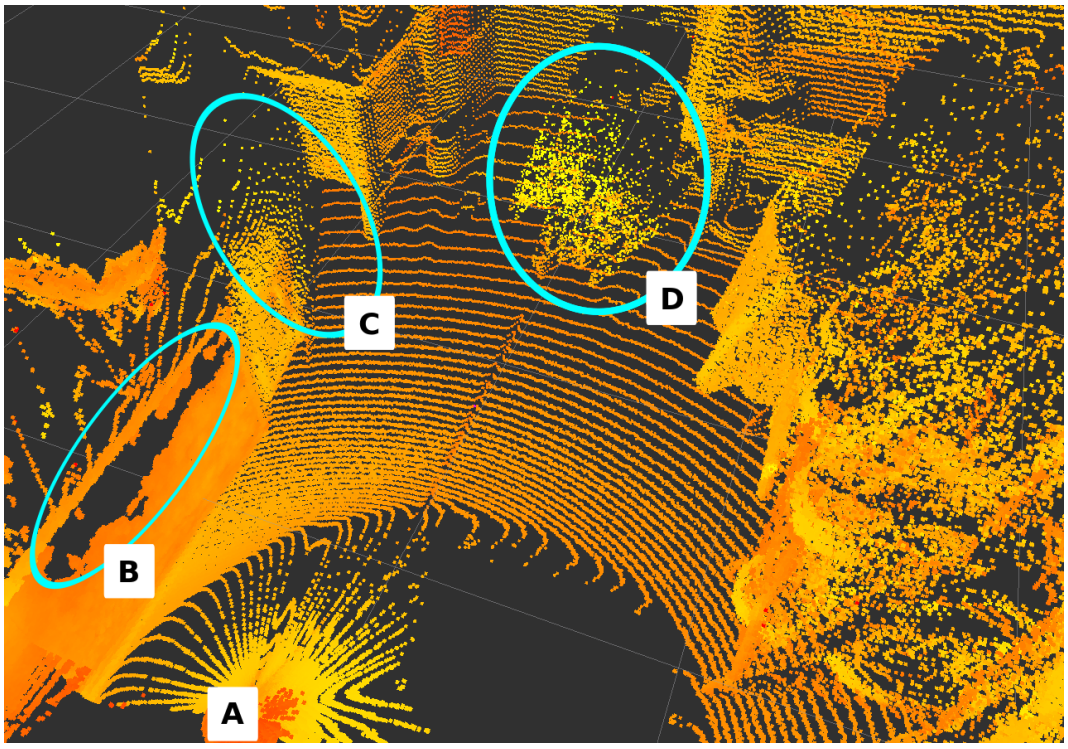
(a)



(b)

Figure 2.2 – Representation of *LiDAR* beams in different conditions along with the resulting waveforms. (a) Different unimodal and multimodal distributions resulting from different target structures. From Jutzi and Stilla [2006]. (b) The multimodal waveform resulting from a laser beam going through multiple translucent material. From the SICK LMS500-21000 Lite website [SICK, b].

(a)



(b)

Figure 2.3 – A picture of a scene (a) and a diagonal view of the point cloud (b) resulting from the acquisition by the SICK LMS151 *LiDAR* mounted on a *PTU*. The sensor position is represented by **A**. Region **B** shows missing points caused by absorbent material (a black box not visible in the picture). Region **C** shows noisy points caused by the edge of an object while region **D** shows a particularly bad *3D* representation of a fine structure (tree branch).

2.3 Data Acquisition

In this section, we report on the relevant characteristics of the four sensors used in our dataset. We then describe the physical configuration of our test setup, then outline the weather conditions pertaining to each of the six collected snowfalls. Finally, we describe how the information from the *LiDAR*s was preprocessed before analysis.

2.3.1 Sensors

Data acquisition was performed with the following four *LiDAR*s: the SICK LMS200, SICK LMS151, Hokuyo UTM-30LX-EW, and the Velodyne HDL-32E. Relevant sensor information is provided in Table 2.1, but the reader is referred to the manufacturers' documentation for additional information[1].

The first element that gives a qualitative overview of the sensor performance is the maximum acquisition distance. This value depends on several factors, such as lighting conditions and target remission. This value is provided directly for the HDL-32E and UTM-30LX-EW, but based on a target remission greater than 75 % for the LMS200 and LMS151. Another element to consider is the shape and area covered by the beam, which influences the probability of hitting a snowflake as well as the proportion of area it covers. A final significant element which changes from one sensor to the other is the number of echoes returned. The Hokuyo sensor can return up to three echoes, which means that it could locate two snowflakes before the beam reaches the ground. Regarding the LMS151, two echoes are evaluated by the hardware, but only one is returned. Finally, note that all *LiDAR*s use class 1 laser with a wavelength of 905 nm.

2.3.2 Setup Configuration

Data acquisition was conducted at Pouliot Hall of Laval University, where sensors were placed close to the inner wall of a window facing N50°E. As shown in Figure 2.4, a wooden structure held the sensors side by side at approximately 14 m above the ground. The main scanning plane (i.e. XY plane in the sensor reference frame) formed a 30° angle with respect to the building wall, so as to increase the maximum distance as much as possible without having the laser beams hitting trees or a

[1] Available here: Velodyne [Velodyne, a], Hokuyo [Hokuyo], LMS151 [SICK, c], LMS200 [SICK, 2006]

Sensor	**Spot shape**	**Spot area** (at 30 m)	**Maximum distance**	**Echoes**
SICK LMS200	Circle	165 cm^2	28 m	1
SICK LMS151	Circle	22 cm^2	50 m	2
Hokuyo UTM-30LX-EW	Ellipse	196 cm^2	30 m	3
Velodyne HDL-32E	Rectangle	51 cm^2	70 m	1

Table 2.1 – Overview of characteristics specific to each *LiDAR*.

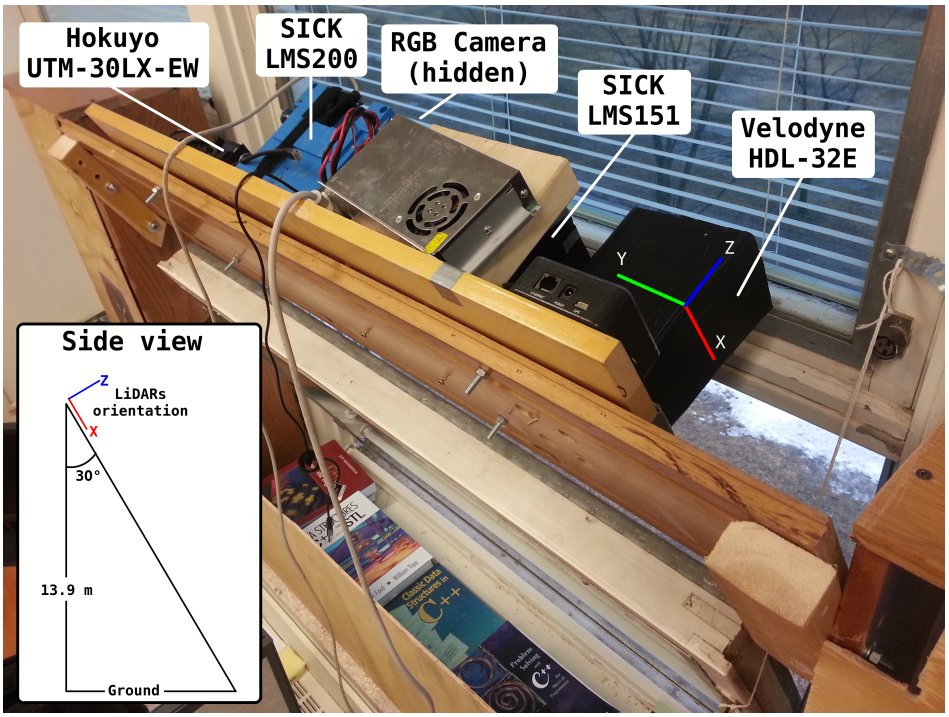


Figure 2.4 – The experimental setup. The *3D* axis represents the orientation of the sensors and the bottom left panel represent the *2D* geometry as seen from the right side of the picture.

pedestrian walkway present near the building. In addition, an *RGB* camera was placed alongside the *LiDAR*s to provide visual information about the scene. In this configuration, a slight opening of the window allowed to keep the instruments inside while scanning outside. To avoid direct interference between sensors, corrugated plastic layers were placed between them. Figure 2.5 shows the scene as observed by the *RGB* camera placed with the sensors.

2.3.3 Dataset Description

Data acquisition started February 12 and ended on March 2. A total of 10 episodes were collected for a total of more than 50 hours of data. Recordings were made using the *Robot Operating System (ROS)*[Willow Garage], which provides standardized data types as well as time synchronization. Data was acquired at different times of day and in a wide variety of conditions, covering a wide range of snowflakes size, falling rate and wind speed. Table 2.2 provides an overview of our data[2]. Of these, six are used in the current study, as highlighted in this table.

2.3.4 Pre-Selection of Laser Data

For each sensor, we selected a combination of angles and laser rings (for the Velodyne) or angles (for the others) that had a clear view of the snow-covered ground surface. The actual details for each

[2]Wind speed, daily precipitation and temperature are reported from Québec City Jean Lesage International Airport at approximately 9 km from Laval University. Data is available here [Government of Canada].

Figure 2.5 – View from the *RGB* camera.

Beginning time	**Duration** (HH:MM)	**Snowflakes size**	**Falling rate**	**Wind speed range** ($km\,h^{-1}$)	**Daily precipitation** (cm)	**Temperature** (°C)
Feb 12, 9:47 am	09:21	Small	Variable	[2–13]	1.4	-14.1
Feb 14, 10:12 pm	04:12	Small	Very low	[5–13]	0.2	-21.4
Feb 19, 8:38 am	10:02	Big/small	High	[3–28]	4.5	-10.9
Mar 2, 1:06 pm	01:27	Big/small	Variable	[22–36]	1.6	-9.1
Mar 3, 10:33 pm	02:17	Big	Medium	[7–9]	5.4	-13.3
Mar 4, 11:45 am	04:12	Big/medium	Low/none	[20–30]	2.0	-4.3
Mar 17, 10:08 am	06:08	Big/medium	Low/none	[1–31]	2.0	-5.8
Mar 21, 6:44 pm	07:42	Medium/big	High	[5–33]	8.6	-5.1
Mar 30, 1:06 pm	04:45	Medium/big	High	[4–8]	8.5	-3.0
Apr 2, 1:56 pm	01:51	Small/rain	High	[2–10]	1.2	-8.4

Table 2.2 – Overview of our snow dataset. Dates in bold correspond to the six days used in the present study.

Sensor	Acquisition frequency	Selected beams/angles	Selected rings	Window size
LMS200	9.375 Hz	55–115	N/A	106 s
LMS151	25.000 Hz	310–220	N/A	40 s
Hokuyo	20.000 Hz	440–590	N/A	100 s
Velodyne	10.000 Hz	-0.05–0.25 rad	17–31	40 s

Table 2.3 – Details of measurement selection for the analysis. The window size is the temporal window used to calculate statistics during the temporal evolution of a snowfall.

sensor are given in Table 2.3. The range of the ground in our scans was between $x = 15\,\text{m}$ to $x = 22\,\text{m}$, depending on the angle. To simplify the analysis, we considered as a snowflake echo any measurement which had a range reading of $x < 14.5\,\text{m}$. As will be shown later in sec. 2.5, this approximation is valid as the vast majority of those events happened for $x < 10\,\text{m}$.

2.4 Temporal Analysis

In this section, we analyze the temporal behaviour of the four sensors for the duration of six complete snowstorms. In particular, we are interested in seeing how the fraction of echoes in snowflakes evolves over time, for all four sensors. First, we will discuss the highly dynamical nature of snowstorms. This will be exemplified by how consecutive scans can have significant quantitative and spatial differences in the distributions of the snowflakes echoes, which justify the use of averaging windows for our analysis. We will then present the actual temporal evolution of these statistics in the form of graphs for all four sensors, and finally briefly discuss the results for each sensor.

2.4.1 Extraction of Temporal Statistics

Snowstorms are highly dynamic processes, with large variation in snowfall rates over their durations. Moreover, the snow physical characteristics (size, shape or reflectance) might vary significantly during a storm, affected by ambient conditions such as humidity level and temperature. Also, wind gusts might pull snow back up in the air or drive it sideways, affecting its effective fall rate. Consequently, one expects during a snowstorm to see significant short, medium and long-term variations in the fraction of *LiDAR* echoes corresponding to the falling snow.

Computing and reporting the temporal statistics for every scan would put too much emphasis on the very short-term statistics. Indeed, the inter-scan variation in the fraction of snowflake echoes can be significant. To better illustrate this point, we have overlaid four consecutive scans in the same plot for the LMS200 and for the first echo returned by the multi-echo Hokuyo sensor in Figure 2.6, for an intense snowing episode from the 02-19 dataset (see Table 2.2). In these figures, we can see strong variations in the fraction of snowflake echoes and their spatial distribution, which we believe can be best described as a random process. One can readily see the fluctuation in these fractions as reported

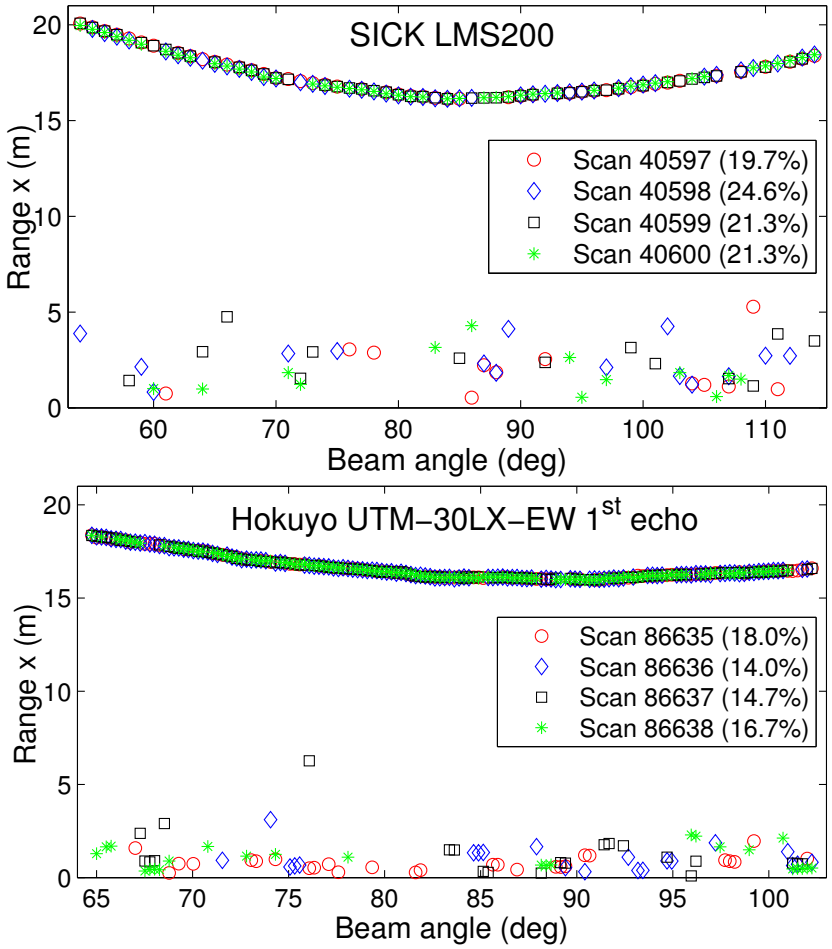


Figure 2.6 – Four overlaid consecutive scans for the LMS200 sensor (top), and the first echo scans for the Hokuyo sensor (bottom), taken from the 02-19 dataset. Each symbol corresponds to a particular scan. The curved line at the top corresponds to the snow surface on the ground. One can see the rapid variation of the snowflake echoes between scans, and how they are mostly limited to a range $x < 5\,\text{m}$. The percentages (in brackets) are the proportion of those echoes in the snowflakes.

in the brackets of the legend in Figure 2.6.

To smooth out these fluctuations, statistics are extracted from a number of consecutive scans contained in a time window of around 1 minute (detailed values in Table 2.3). Figure 2.7 shows this smoothed fraction of snowflake echoes compared to all returned laser measurements as a function of time, for the six snowiest days of our dataset. To allow for better visualization, only the LMS200 and the Hokuyo's first echo are plotted at their actual scale (1x): Others have been scaled up (from 30x to 200x), with their corresponding scaling factors reported in the legend. As will be shown below, some sensors were much more sensitive than others.

2.4.2 Detailed Analysis, per Sensor

SICK Sensors LMS200 and LMS151

Our first conclusion based on Figure 2.7 is that the most sensitive device was the older LMS200, first introduced in the mid-2000s. For the most intense snowstorms (Figure 2.7. b) 02-19, d) 03-17, e) 03-21 and f) 03-30), it peaked at around 15% of measurements triggered by the falling snowflakes, for averaging windows of 106 s. As an older-generation device, it probably uses less sophisticated algorithms and sensing, and was not directly targeted for harsh outdoor environments. Indeed, its technical description [SICK, 2006] indicates that "Raindrops and snow-flakes are cut out using pixel-oriented evaluation", but this seems only applicable to obstacle detection (field computation), not the actual measurements. No further details are given. On the other hand, the more recent SICK LMS151 exhibits much less sensitivity to snowflakes: The reduction factor for the fraction of snowflakes echoes is in the order of 200-300, granting this device a much higher immunity in snowstorms. Indeed, the highest peak was around 0.1 % of echoes in snowflakes during the 02-19 dataset. In some sense, this is not surprising considering that the documentation from the manufacturer mentions that this model is targeted for "all-weather conditions" [SICK, a].

Hokuyo UTM-30LX-EW

For this sensor, we resorted to a slightly different approach for comparison, as the device has been designed to return multiple echoes. We thus extracted statistics for the two most relevant cases: for first and last echoes. Statistics for the first echo tell us how sensitive the device is, if one wishes to detect the presence or absence of falling snow. This information could be used, in turn, to adapt the driving strategy of an autonomous vehicle or inform vision algorithms of the presence of particles in the air. On the other hand, using the last echo increases the probability that we will detect obstacles, such as another vehicle or the snow-covered ground. This information would be used for localization and navigation purposes. In the case of the first echo, we observed that the device behaved similarly to the LMS200. Indeed, the Hokuyo first echo (blue line) closely tracks the LMS200 curves (red dashed line) almost everywhere in Figure 2.7, with a few exceptions. In the case where we look at the last echo, the sensor behaves like the LMS151, not surprisingly as this sensor does a 2-echoes analysis and filtering. The last echo of the Hokuyo tends to reject the falling snow, but not as well as the LMS151, as it peaked at around 0.5 % in some episodes. Nevertheless, this difference might not be sufficient to impact algorithms relying on laser data. Note that Table 2.4 shows similar correlations between these three sensors, for the averages taken over the complete 02-19 dataset.

Velodyne HDL-32E

For all purposes, the behaviour of the Velodyne was similar to the last echo of the Hokuyo sensor. This is seen both in the temporal behaviour in Figure 2.7 and in the average value displayed in Table 2.4.

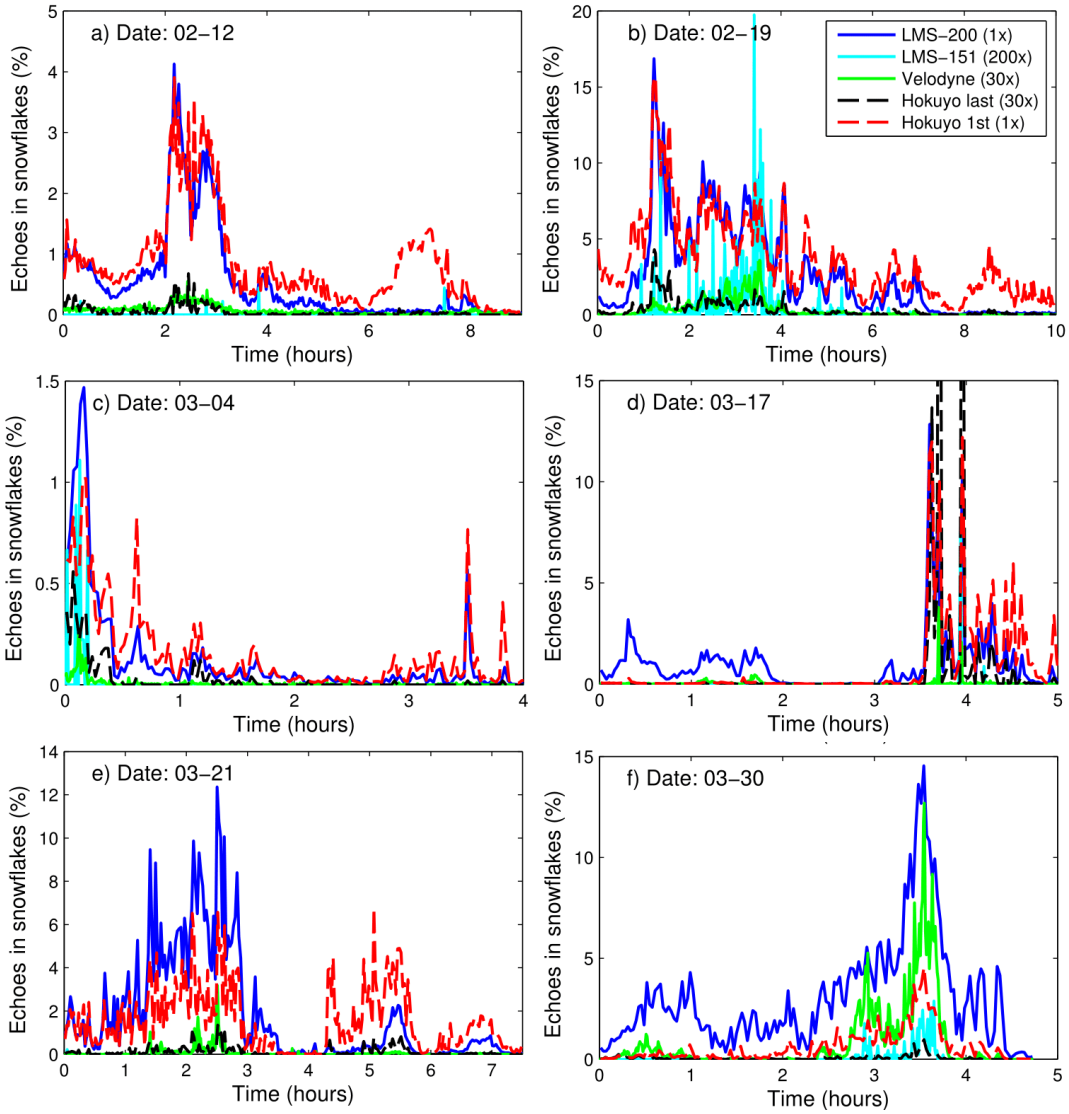


Figure 2.7 – Temporal evolution of the percentage of echoes coming from the falling snow (range x <5 m) during the 6 most intense episodes, for all 4 sensors. The data is smoothed by taking statistics for small-time windows. Except for the LMS200 and Hokuyo first echo, all other sensors statistics have been scaled up (factor in brackets of legend b) for ease of visual comparison. Time is in hours, starting from the beginning of the data capture sequence.

LMS200	Hokuyo first echo	Hokuyo last echo	LMS151	Velodyne HDL-32E
2.67%	3.55%	0.0113%	0.00178%	0.0100%

Table 2.4 – Overall average snowflake echoes for the complete 02-19 dataset, per sensor. These averages are significantly lower than the instantaneous values displayed in Figure 2.7, as snow was not falling at all times during that period.

2.5 Distribution of Snowflake Echoes as a Function of Range

In the previous section, we showed how the expected fraction of snowflake echoes varied temporally during snowstorms. In some sense, it provided for a *temporal* modelling of the interaction between a snowstorm and a given *LiDAR*. In this section, we evacuate the temporal aspect and instead focus on how the range x affects the probability for a snowflake to trigger a measurement. To this end, we will use histograms to estimate a probability density function of those events, and show that for the weather conditions and the sensors we tested, there seems to be an upper bound on the range x beyond which falling snowflakes no longer trigger a measurement: In other words, snowflakes become invisible to the sensor past a certain range.

2.5.1 Modelling the Impact of Range on Snowflake Detection

When modelling a range sensor, one has to have an idea of the probability distribution of certain events (e.g. snowflakes) as a function of this range. Over the years, many researchers have proposed probabilistic models for sensors, notably in Burgard et al. [2006]. In the previous section, we have in some sense estimated the probability for a given sensor S that a snowflake would generate an echo $E_{\text{snowflake}}$ given the weather conditions W, or $P_S(E_{\text{snowflake}}|W)$. In this section, we take a closer look at which range x such events would be generated, that is $P_S(E_{\text{snowflake}}|x,W)$. Having such a formulation would allow for a more statistically-sound treatment of the information, such as within a Bayesian probabilistic framework. To this effect, we use histograms as approximations to the previous distribution. In Figure 2.9, we have plotted these histograms for each of the four sensors. For ease of comparison, they have all been normalized by their total area in the interval $0 < x < 14\,\text{m}$, as the total count varies widely between the sensors. The numbers in brackets in the legend indicate the fraction of echoes in the snowflakes compared to the total number of data points, for a given dataset.

The general shape of these histograms is close to a log-normal distribution, with the exception of the LMS200 for a number of dates (02-12 through 03-17), which seems to follow a sum of two log-normal distributions. We attribute this log-normal shape to the interaction between two different phenomena, illustrated in a cartoon-type model in Figure 2.8. At short ranges $x < 3\,\text{m}$, the building acts as a shield and decreases the probability of having a snowflake in the path of the laser. We recognize that this phenomenon would be most likely absent on an autonomous vehicle, thereby increasing the probability of having echoes in snowflakes at close range. However, we believe that this difference is not problematic, as close obstacles would be easily detected from *i)* the overwhelming number of *LiDAR* echoes on this obstacle *ii)* other sensing modalities such as vision or radar. Furthermore, if the *LiDAR* is to be mounted on a rooftop, one can safely ignore echoes in the first 2 m, either in software or directly through the sensor itself (via its configuration). The other phenomenon, illustrated as the red dashed line in Figure 2.8, is the probability of optical detection of a snowflake by the sensor as a function of the range x. We argue that this shape is due to the rapidly decreasing light intensity of the echoes in snowflakes, as a function of x. Combining these two phenomena yields a log-normal shaped curve (black line in Figure 2.8). Overall, this seems to indicate that a simple probabilistic

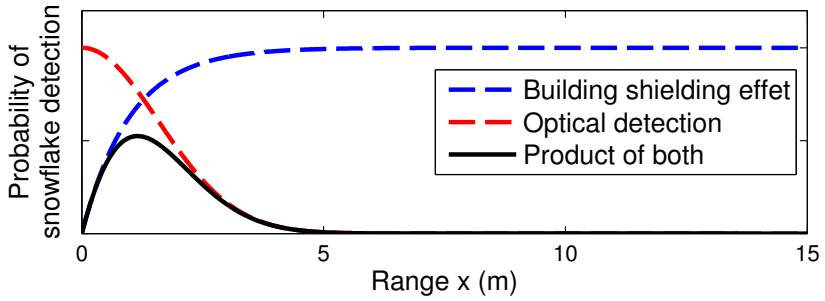


Figure 2.8 – Cartoon representation of the interaction between the probability of detecting a snowflake (in red) and the diminution of snowflakes due to the shielding effect of the building (in blue). The black line is the product of the two, and bear a close resemblance to the actual histograms extracted from our dataset.

model $P_S(E_{\text{snowflake}}|x,W)$ can be derived for these sensors.

2.5.2 Sensor Results

As can be seen from the histograms in Figure 2.9, most sensors exhibit the log-normal or sum-of-log-normal distributions discussed above. We note that for certain days, the distributions are shifted to the right (greater range x). In particular, for the 03-21 and the 03-30 distributions, this shift is substantial (on the order of 1 m). We suspect that for these days, the snowflakes were significantly larger, thus allowing for a stronger optical echo and extended range of detection.

For all sensors, we can also conclude that beyond the range $x > 10\,\text{m}$, snowflakes are no longer detected, i.e. they become invisible. A small notable exception would be for the Velodyne, for which snowflakes were detected all the way to $x = 14\,\text{m}$, albeit at a significantly reduced rate. Again, we do not think that this would significantly impair their use in conditions similar to our test setup.

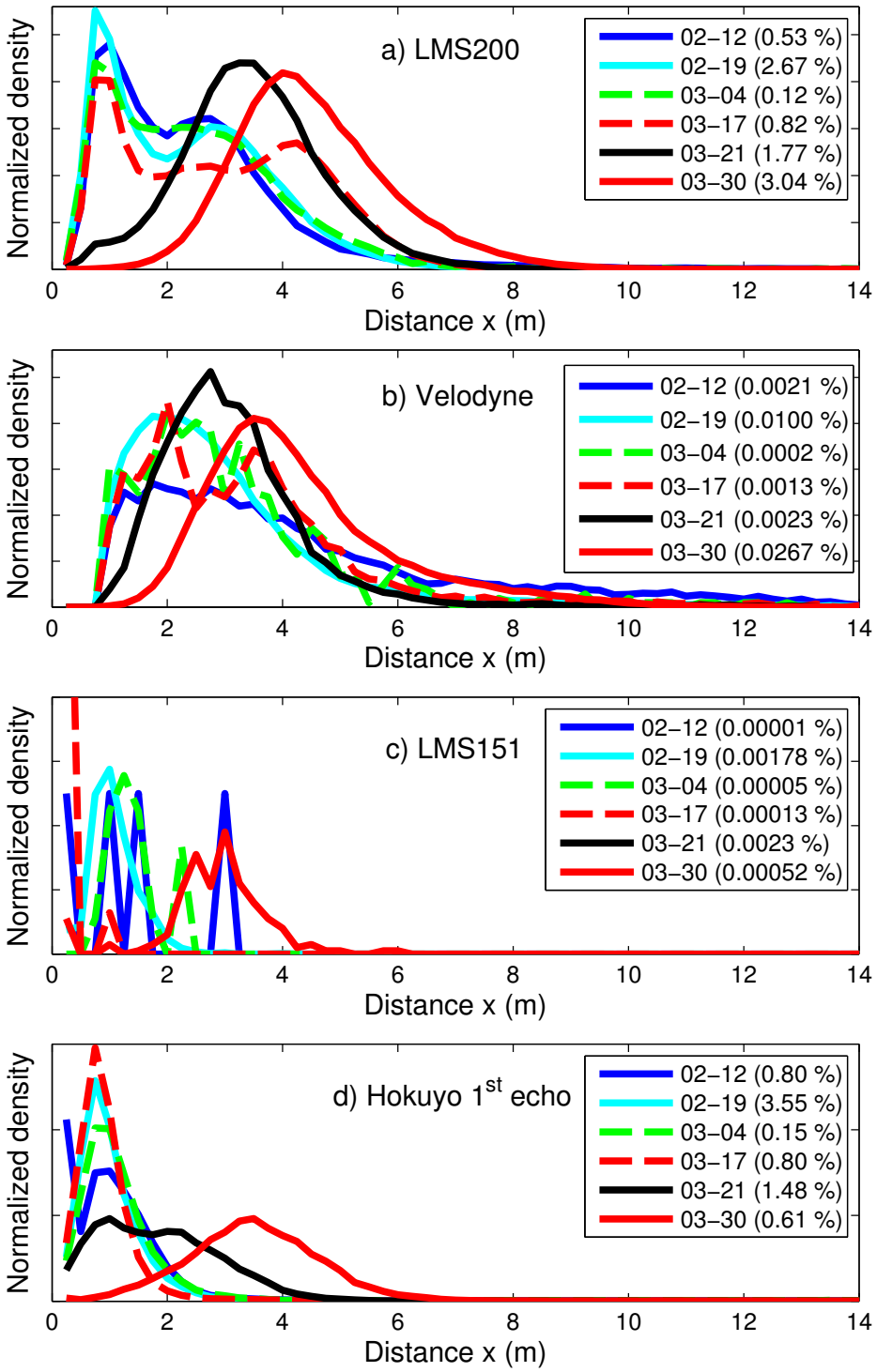


Figure 2.9 – Histograms of echoes in falling snow during important snowfall days, as a function of distance x reported by the sensor. Each histogram has been normalized by its area, for ease of comparison. The numbers in brackets are the fraction of data points in the complete dataset that correspond to snowflake echoes. Note that for the 03-21 dataset, the LMS151 was not working properly: thus no data is included for that day.

2.6 Discussion and Conclusion

In this chapter, we explored the impact of falling snow on the usability of 4 commonly deployed *LiDAR*s. To this end, we first presented an overview of *LiDAR*s functioning and possible causes of noise in the measurements. As explained, the small size and dynamic nature of snowflakes make snowstorms perfect examples of challenging condition when using *LiDAR*s.

For our experiments, we collected data during 6 snowstorms in the winter of 2015. Upon analysis, we found that the SICK LMS200 was the most sensitive *LiDAR*, having a peak average rate of up to 15 % of echoes coming from falling snow. Meanwhile, all three others never exceeded 1 %. We also presented a simple probabilistic model to take into account the effect of the range on snowflakes interference. Based on a histogram analysis, we concluded that for our experimental setup, this model can be approximated by a log-normal distribution. Most importantly, our data indicate that the impact of snowflakes on *LiDAR* beyond a range of 10 m is very limited.

However, a number of questions remains to explore. For example, as the *LiDAR* beam travels through the falling snow, its intensity will diminish. Since the maximum range of a *LiDAR* is heavily related to this beam intensity, we expect the maximum range to be affected during snowstorms. In our setup, we have not witnessed this issue, indicating that this effect probably happens beyond our maximum distance of 20 m. Another aspect to be investigated is the relationship between the returned intensities and the surface type (ground or snowflakes). Also, because of the shielding effect of the building, very few snowflakes were present at close range; It might be the case that at closer range, a snowflake might be detected at more than one angle, effectively occluding small targets. Moreover, we have not investigated the impact on the measurement noise for the snowy ground surface in the presence of falling snow.

Chapter 3

Forest Influence on LiDAR-Based Place Recognition

3.1 Introduction

In order to navigate safely and efficiently in their environments, mobile robots have to be able to solve a multitude of problems. An example of such problems is the ability to determine whether the robot is located in a place it visited before or in a new, unvisited one. Despite the fact that this question seems relatively elementary, solving the so-called place recognition problem is useful for a wide variety of applications. For instance, multiple robots can cooperate to concurrently build a global map (multi-session mapping) using recognized places as connecting points between the different local maps [Howard, 2004]. The well-known "kidnapped robot" problem, which consists in determining if the robot has been carried to an arbitrary location, can also be solved using a place recognition algorithm. Because such algorithms do not rely on odometry and allow a robot to locate itself relative to all previously visited places, it is possible to detect this kind of unpredictable change of location. Finally, place recognition algorithms are useful to perform *SLAM*. The most obvious use is for a topological representation, where the map consists of places and links between them, but it is also essential for loop closure (often referred as the "front-end") when using a metric representation.

In the previous chapter, we analyzed the influence of an environment with snowy condition on the *LiDAR* data. Following the same idea, we are now interested identifying the impact of challenging environments when performing place recognition using *LiDAR*. More precisely, we want to evaluate how unstructured environments, such as forests, influence navigation algorithms performance. To this end, we chose to use a state-of-the-art *LiDAR*-based place recognition algorithm developed by Steder et al. [2011b]. Their algorithm proved to be successful in structured indoor and semi-structured outdoor environments. In our experiments, we produced our own datasets in forests, but also in semi-structured conditions for comparison purposes. Besides the influence of the type of environment, we are also interested in the impact of the sensor used and data associated with it. For this analysis, we used

Device	Manufacturer	Model	Use
Computer	–	–	Data acquisition/synchronization
Gateway	Microhard System Inc.	VIP2400	Network and Wi-Fi communication
Gamepad	Logitech	F710	Remote control of the robot movements
LiDAR	Velodyne	HDL-32E	Point cloud acquisition
LiDAR	SICK	LMS151	Point cloud acquisition
PTU	FLIR Motion Control Systems	D46-17	Rotate the SICK *LiDAR*
IMU	ChRobotics	UM6	Odometry
Camera	Axis	M1013	Visual reference
GPS	NovAtel	SMART6	Not used

Table 3.1 – List of devices available on the Husky A200 and their use in our experiments. Note that both *LiDAR*s were never mounted at the same time.

two sensors, namely the SICK LMS151 and the Velodyne HDL-32E. The chapter is divided as follows, Section 3.2 details where and how the datasets were produced, as well as the resulting data. Thereafter, fundamental concepts related to the place recognition algorithm and the algorithm itself will be presented in Section 3.3. Finally, the results of the comparative analysis will be presented in Section 3.4 before we conclude.

3.2 Data Acquisition

In the following section, we first describe the robotic platform and sensors used to gather our place recognition dataset. We then describe in more details the dataset acquisition procedure and the resulting data.

3.2.1 Robotic Platform

The Husky A200 is a medium size ($990\,\text{mm} \times 670\,\text{mm} \times 390\,\text{mm}$) *UGV* developed by Clearpath Robotics. It is a rugged robot designed for all terrain conditions and it uses a differential-drive skid steer, allowing easy control and in-place turns. The maximum speed of the vehicle is $1\,\text{m}\,\text{s}^{-1}$ and the maximum payload capability is 75 kg. This robotic platform is well suited for our needs, because it can move in forests and carry the required equipment. Our platform is shown in Figure 3.1 and details about the available devices are presented in Table 3.1.

The on-board computer (2.4 GHz Intel i5-520M) is an essential element of our experiments, as it connects all devices, acts as a control interface for the robot and stores the acquired data. The computer does not provide a *Graphical User Interface (GUI)*, but is connected to the gateway that broadcasts a WiFi network, allowing *Secure Shell (SSH)* communication. The platform is also equipped with a wireless gamepad, which enables manual control of the movements of the robot. Point clouds

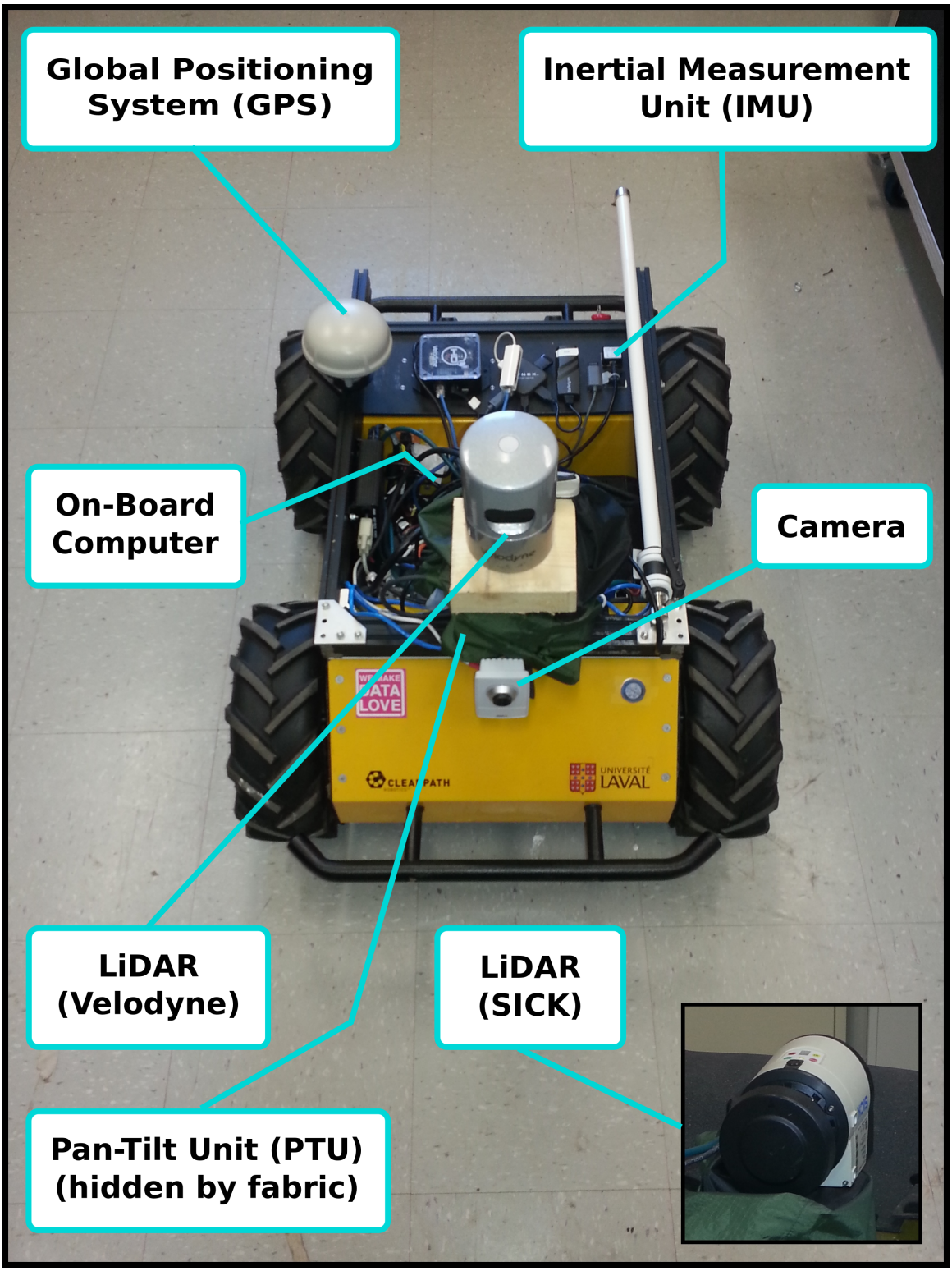


Figure 3.1 – Our robotic platform (Husky A200) and the devices used for data acquisition. The SICK *LiDAR* is not mounted, but instead is shown on the bottom right corner of the figure. Note that the *PTU* is hidden by a cover and the on-board computer is mostly occluded by the Velodyne *LiDAR*.

acquisition is possible using either the SICK LMS151 or the Velodyne HDL-32E. The selected sensor is mounted on the *PTU*, which remains fixed for the Velodyne, but is rotated with the SICK to merge multiple *2D* scans into a single *3D* point cloud. Section 3.2.2 gives more details about the resulting point clouds for each sensor.

The sensors described above are essential for our place recognition research, but we also use the wheel encoders along with the *IMU* for odometry estimation and the camera for visual reference of the dataset. Note that we do not use the *GPS*, as it is not reliable in forests.

3.2.2 Dataset Description

Data acquisition was performed using *ROS*, a set of software libraries and tools created to simplify the development of robotics applications. It provides, out of the box, all drivers for the Husky and our sensors. Its data publishing system provides timestamps that allow easy synchronization between sensors. The recording tool (rosbag) was used to create our datasets, with data processing performed a posteriori.

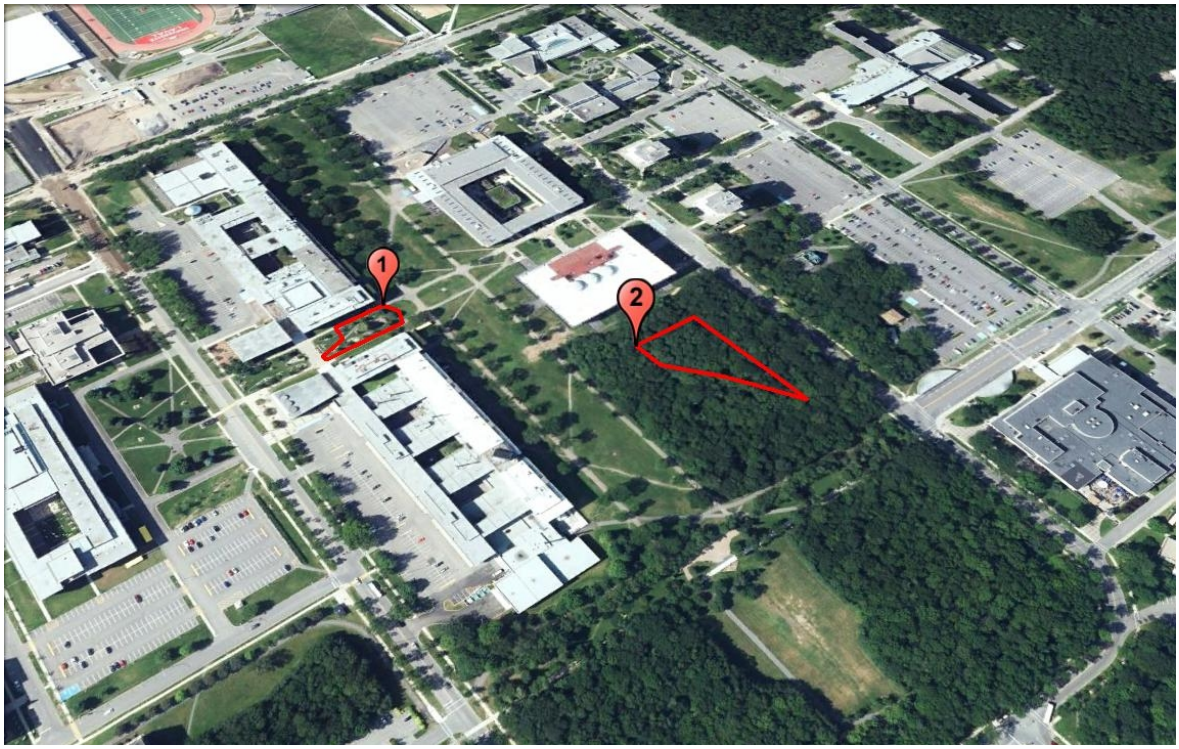
We produced datasets in two different areas of the Laval University campus. An aerial overview of the path followed by the robot at these two locations is presented in Figure 3.2.

The first site was chosen for its more structured nature and is located between the Alexandre-Vachon and the Adrien-Pouliot buildings. This environment is mostly open, the terrain is smooth and flat and the site contains man-made objects such as buildings, stairs or tables. Examples of pictures acquired by the robot on this site are presented in Figure 3.3a and 3.3b. This dataset closely resembles the kind of data on which several place recognition algorithms are typically tested on (e.g. Freiburg Campus 360 degrees *3D* scans [Universität Freiburg], Robotic *3D* Scan Repository [Nüchter and Lingemann]).

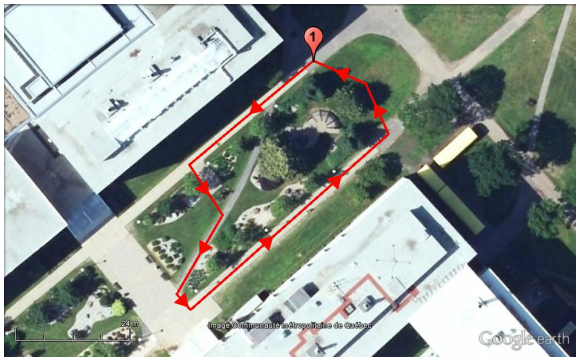
The second site, chosen for its unstructured nature, is located in a wooded area, also on the Laval University campus. The dataset path starts on a pedestrian walkway, but after its second turn (of approximately 330°), it continues for around 100 m in rougher terrain. This forested environment presents multiple small structures and significant of occlusions. Figure 3.3c and Figure 3.3d show pictures from the robot camera at this location. This dataset will allow us to better understand the influence of a less structured environment on the place recognition algorithm. In particular, the absence of large flat surfaces and corners typical of buildings, as well as the closeness of the space, will be challenging for place recognition algorithms.

To evaluate the impact of the sensor used and data associated with it, we will use the SICK and the Velodyne, for which you can find details in Table 3.2.

The SICK is a *2D LiDAR* with a scanning angle of 270°, a resolution of 0.5° and an acquisition frequency of 50 Hz. This sensor was placed on the *PTU* so that the blind spot faced downward. During acquisition, it was rotated around the vertical axis (pan) at a speed of $14.32\,^{\circ}\,s^{-1}$ for half a turn, while the vehicle was stationary. This procedure allowed the acquisition of 628 *2D* scans of



(a)



(b)

(c)

Figure 3.2 – (a) Partial aerial view of the Laval University campus including the two approximate paths followed by the robot for data acquisition. (b) Zoomed view of the path followed (counterclockwise from tag 1) in a structured environment. The length of this path is approximately 160 m. (c) Zoomed view of the path followed (clockwise from tag 2) to create the unstructured datasets. The length of this path is approximately 275 m. Images source: Google Earth, (2015)

(a) (b) (c) (d)

Figure 3.3 – Images from the *UGV* camera during the acquisition of the structured dataset (a) (b) and the unstructured dataset (c) (d). Note that the images are at an angle for the unstructured dataset, because the camera was not aligned with the robot during acquisition.

540 points, later merged into a single *3D* point cloud. To create the dataset using the SICK, the robot was stopped at regular intervals on the established path to acquire scans.

The Velodyne, for its part, directly allows *3D* point cloud acquisition by spinning 32 lasers around its vertical axis. These lasers are evenly distributed between $-30.67°$ and $10.67°$ relative to the horizontal plane. According to the Velodyne datasheet [Velodyne, b], the device acquires approximately 700 000 points s^{-1} and publishes at 10 Hz, therefore creating point clouds of 70 000 points. Note that these point counts are variable as the rotation speed can change slightly and the use of the *User Datagram Protocol (UDP)* can lead to some loss of points. Regarding the dataset acquisition, the robot was driven at a constant speed ($0.3\,m\,s^{-1}$) and performed data acquisition simultaneously. In order to obtain a quantity of scans similar to that produced with the SICK, only one out of 80 scans was used to create the final dataset.

Table 3.2 summarizes information about the sensors resolution and *Field Of View (FOV)*, while Table 3.3 provides a name for later references and additional information about each of our datasets.

Sensor	Horizontal resolution	Vertical resolution	Minimum angle	Maximum angle	Point counts
SICK LMS151	0.57°	0.5°	−45.00°	90.00°	339120
Velodyne HDL-32E	0.16°	1.33°	−30.68°	10.67°	72000

Table 3.2 – Details about the point clouds created with our two *LiDAR*s. The minimum and maximum angles are given relative to an horizontal plane in the sensor frame of reference and both sensors report 360° around the vertical axis. The point counts represent the maximum number of points in the resulting point cloud.

Dataset name	Site	Sensor	Date	N_{L1}	N_{L2}
`Structured-SICK`	Structured	SICK LMS151	July 16th, 2015	81	83
`Unstructured-SICK`	Unstructured	SICK LMS151	July 14th, 2015	94	92
`Unstructured-Velodyne`	Unstructured	Velodyne HDL-32E	May 28th, 2015	104	104

Table 3.3 – Datasets acquired for place recognition analysis. We define a name for each dataset in order to facilitate reference thereto in the remainder of this document. N_{L1} and N_{L2} represent the number of scans acquired during the first and second loop respectively.

Examples of point clouds from our datasets are shown in Figure 3.4. We observe that the unstructured environment is more congested and objects are closer to the sensor than for the structured environment. In fact, the average distance of points from the sensor in the unstructured environment is 7.61 m compared to 9.23 m for the structured environment. There is also more space without obstacles in the sensor range for the structured dataset. The percentage of points return using the SICK is 82.6 % on average in the unstructured dataset compared to 40.2 % in the structured dataset. These point clouds also illustrate the higher vertical resolution and larger *FOV* of the SICK compared to the Velodyne. The influence of these factors will be discussed in Section 3.4.

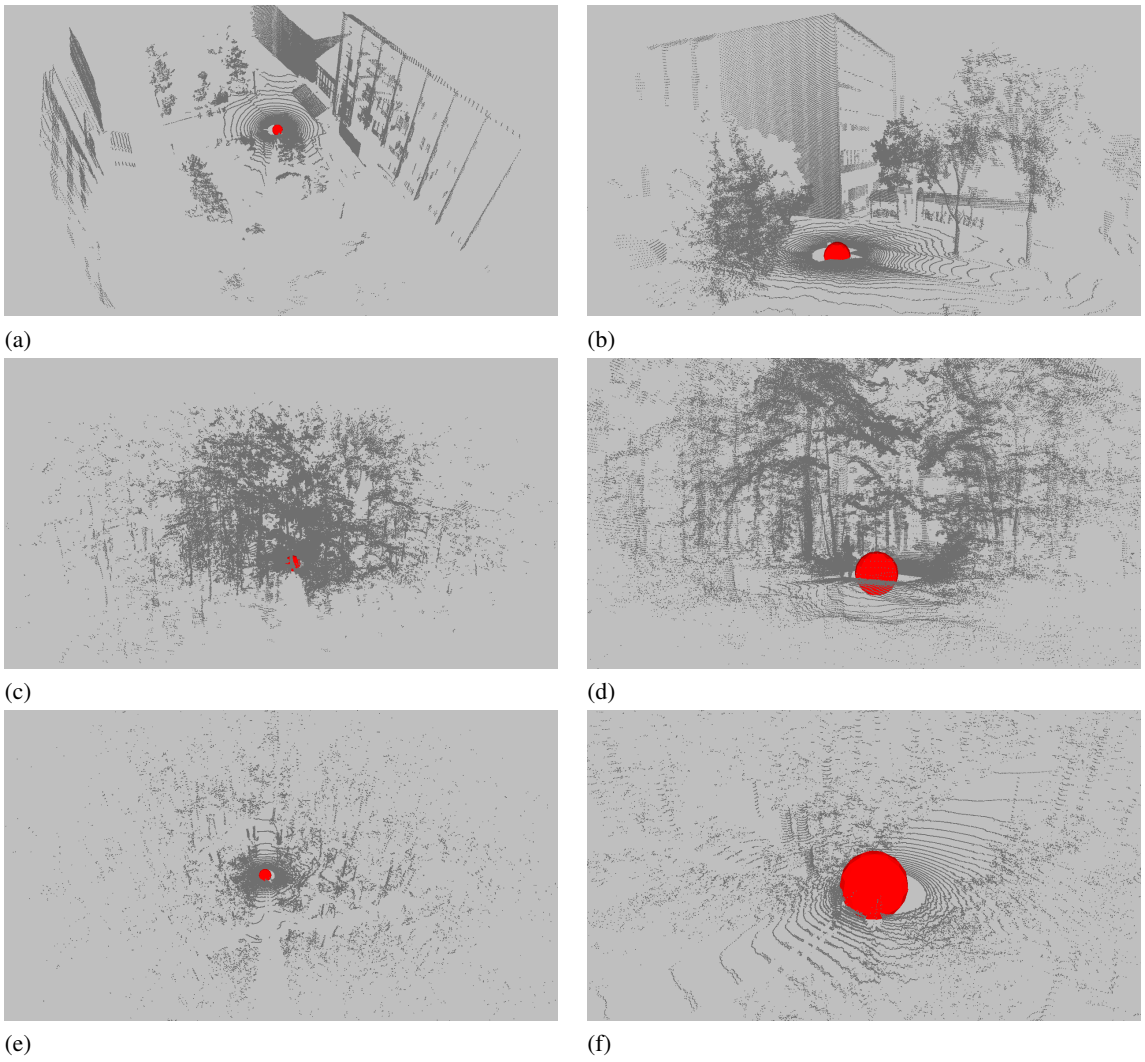


Figure 3.4 – Examples of point clouds from our datasets viewed from different perspectives. (a) and (b) show acquisition from the SICK in the structured environment. (c) and (d) also show point clouds from the SICK, but in the unstructured environment. Finally, (e) and (f) show the result of acquisition from the Velodyne in the unstructured environment. Robot position is represented by a red sphere with a 1 m radius in each picture.

3.3 Place Recognition Algorithm

While there are a number of articles presenting place recognition algorithms using different sensors (e.g. camera [Cummins and Newman, 2011], stereo-camera [Cadena et al., 2012]), the literature of such algorithms based solely on *3D* data is limited. For our place recognition analysis, we choose to evaluate on our datasets the state-of-the-art algorithm (at the time of the experiments) developed by Steder et al. [2011b] on our datasets. We will first describe some fundamental concepts used for place recognition in Section 3.3.1 and then we will give an overview of the Steder algorithm itself in Section 3.3.2. For the interested reader, the full details of the algorithm are available in [Steder et al., 2011b].

3.3.1 Fundamental Concepts

The following subsection is an introduction to basic concepts for understanding *3D* place recognition algorithms in general. In particular, conventional methods for representing and comparing *3D* acquisitions will be presented.

Feature Keypoints and Descriptors

The first step in determining whether or not a place has been visited before, is to convert the sensor data into a format that is more convenient for identification. The generally adopted representation is a vector of real numbers, called a descriptor. This mathematical representation aims at being more compact than the original data while trying to capture significant characteristics from a recognition point of view.

A descriptor can be *global*, meaning that it tries to capture information about the whole scan, or *local*, meaning that it does the same but only for a specific subregion of the scan. When using local descriptors, one needs to select the keypoints around which the descriptors will be extracted. These keypoints can be at pre-determined and fixed locations (e.g. division in a simple grid) or selected using more refined algorithms. A common practice for the latter is to choose keypoints in so-called interesting regions which are often defined as regions of high gradient (e.g. edges, corners), as these regions generally contains more information than smooth surfaces. Note that, for simplicity, we might use the term **features** as a more general term for keypoints and their respective descriptors in the remainder of this document.

The concepts of keypoints and descriptors first originated from the computer vision literature. More recently, they have been adapted for *3D* data. Some popular examples of features for both types of data are shown in Table 1.1. Note that some algorithms propose solutions for both keypoints detection and descriptors (e.g. SIFT, NARF). However it is not mandatory to use them together as any combination is generally valid. While all features are different, they were all developed with the same goals in mind:

- Distinctiveness: each feature should be easily differentiable with respect to others.
- Repeatability: the feature values should be stable under changes including:
    - Transformations: rigid transformation for point clouds and projective transformation for images, but also changes in the pose of the objects and/or the viewpoint.
    - Noise: small variations in measurements (range/intensity) and occasional erroneous values (points/pixels).
    - Resolution: the number of points or pixels representing a given area.

Range Image and NARF Feature

In this subsection, we present the NARF keypoints and descriptors, as well as the *3D* data representation on which they rely: the range image. We will see that the place recognition algorithm, presented in Section 3.3.2, relies on this *3D* representation to determine a matching score between scans.

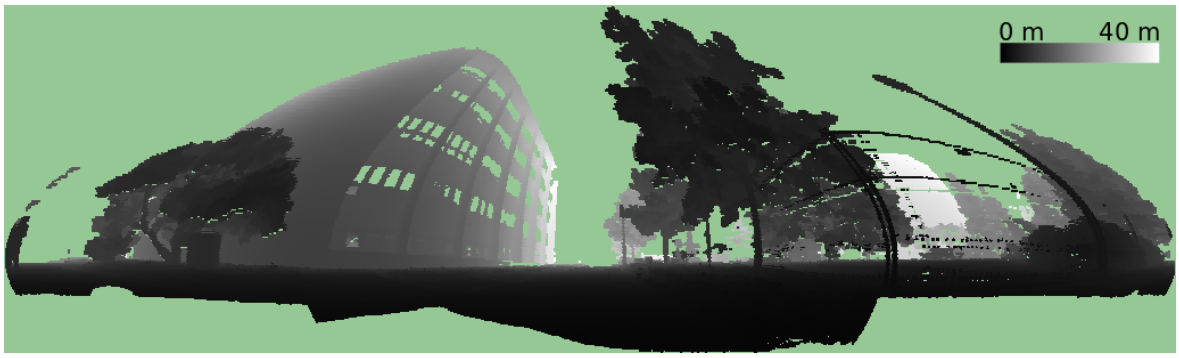
Range images represent a *3D* data point by a pixel position (i.e x and y) and a range value. Note that the position of the pixel actually represents a horizontal and vertical angular position. For an omnidirectional *LiDAR* scan, the corresponding range image is a spherical projection of the points from the centre of the sensor. Range images can be better defined by the constraints they must meet to be valid.

Firstly, converting a *3D* scan into a range image requires the acquisition to originate from a single view point and to have a single range value per pixel. It is therefore not possible to use point clouds acquired with multi-echoes *LiDAR*s or produced by merging multiple scans. Fortunately, the latter constraints are met in our datasets.

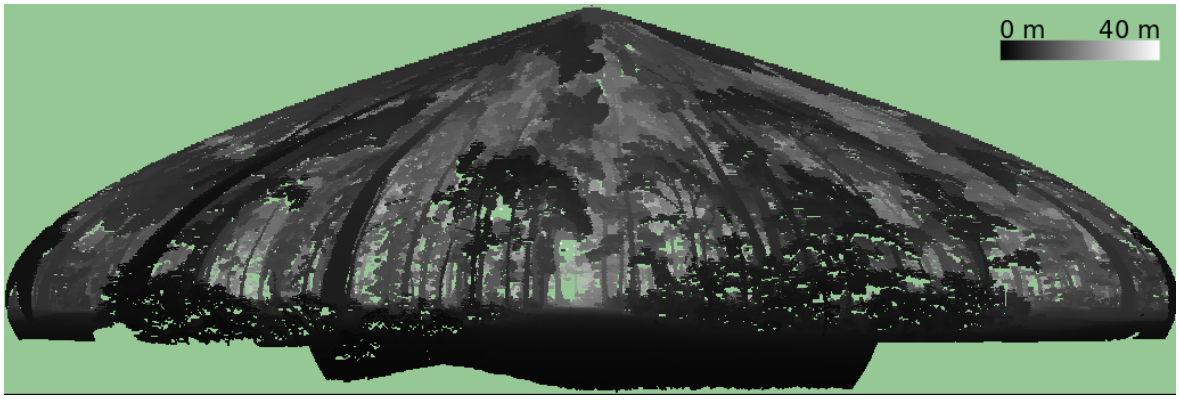
Secondly, every pixel in a range image must have a value. Since *LiDAR*s have a minimal and a maximal range, the data points outside this interval are ignored. Similarly, when a laser beam hits a highly absorbing or a reflecting surface at an angle, the sensor will not get any return. Such missing data points cause pixels of the range image to have no value and are instead considered as far range (i.e. the maximum range of the sensor).

Finally, the resolution has to be adjusted so that each pixel of the resulting range image covers the same angular resolution vertically and horizontally. Images being discretized in pixels, the values of these can either be a weighted average of the laser beam ranges, or the value of the smaller range within his area, depending on the user preference. The converted scans have a dense and uniform representation similar to grayscale images, allowing the reuse of standard image processing techniques. Examples of range images can be seen in Figure 3.5.

As indicated previously, feature keypoints are generally chosen in high gradient regions of the *3D* inputs. A problem arises when the input data is in the form of a point cloud, as it is impossible to distinguish edges caused by object boundaries from those caused by occlusion. Figure 3.6 present an example of a partial range image and the corresponding point cloud, where you can see edges caused



(a)

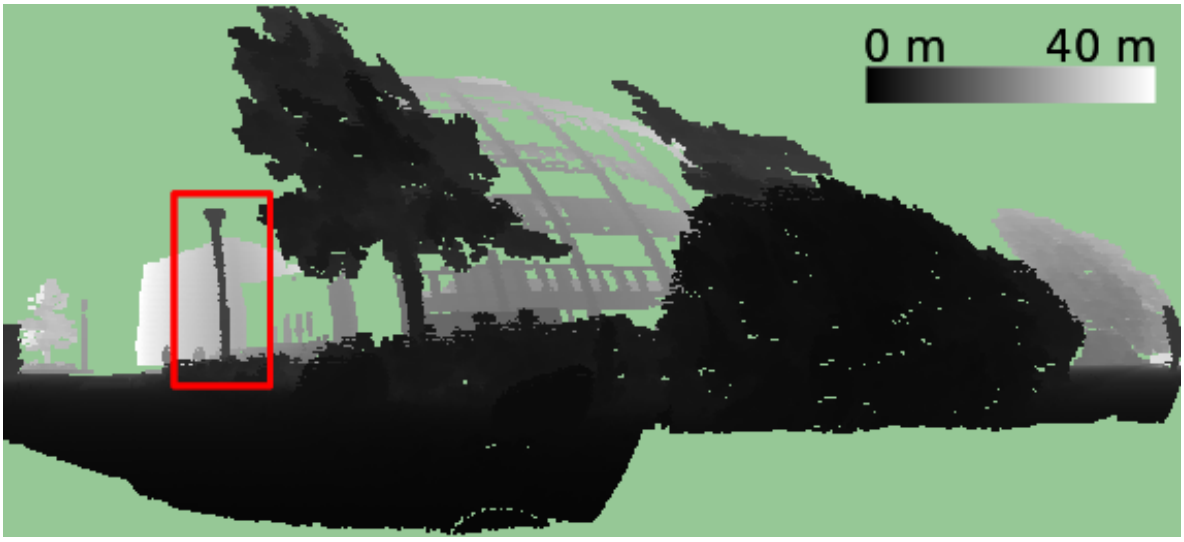


(b)

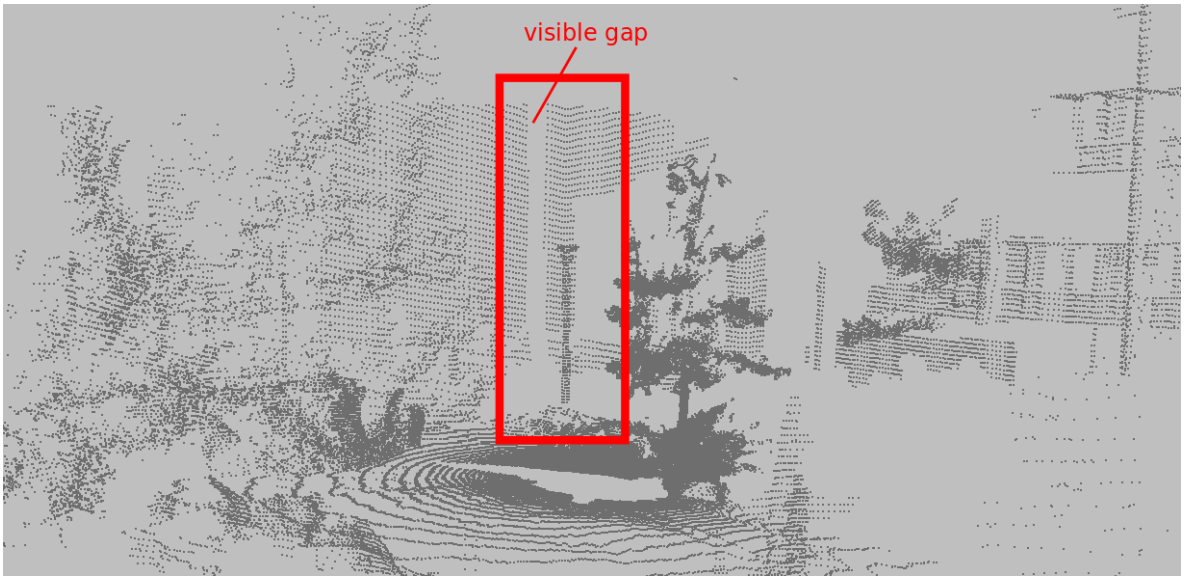
Figure 3.5 – Examples of range images for the `Structured-SICK` dataset (a) and the `Unstructured-SICK` dataset (b). Note that objects appear distorted due to the projection on the plane. The background colour (light green) corresponds to areas without laser return. All other pixels represent ranges between 0 m (black) and 40 m (white).

by an object boundary and edges caused by the occlusion of this object. Edges caused by occlusions can generate meaningless keypoints and the descriptors built based on these keypoints leads to a poor representation of the environment, which can reduce the place recognition performance. This is particularly useful in our context of complex environments such as forest, because there are significant occlusions. The article describing the NARF features [Steder et al., 2011a] details how using range images allow to differentiate these two types of edges.

The last item to be discussed concerning the NARF features is the descriptor structure. In order to create this NARF descriptor, one must first compute the normal to the surface at the keypoint location. A star pattern, as shown in Figure 3.7, is then overlaid on top of a small range image patch, as seen by an observer looking at the keypoint along the normal. Each beam of the star pattern corresponds to a single value in the final descriptor. This value captures how much the pixels changes under the beam (i.e. its gradient). A unique orientation around the normal is finally determined, to ensure that the NARF descriptor is rotation invariant. This orientation corresponds to that of the beam having the largest value. An interesting characteristic of the NARF descriptors is that each of them encodes its own local coordinate system, allowing for a complete six degrees of freedom pose identification.



(a)



(b)

Figure 3.6 – Example of a partial range image (a) and the corresponding point cloud (b). The red rectangle highlight a region of the environment where, from the robot point of view, a lamp post partially occludes a building wall. In the point cloud, this configuration leads to edges on both the lamp post boundary and the wall itself. In reality, the information contained in this scan does not allow to know what is behind the post and the edges of the wall are most likely artifacts caused by occlusion. In contrary, the range image is built taking into account the position of the sensor, thus allowing to consider only the boundary of the lamp post as edges and ignoring edges on the wall. Indeed, one can ignore the gap in the wall for the range image, as it is readily explained by the occlusion of the range image. Determining this occlusion is not possible by nature in a point cloud.

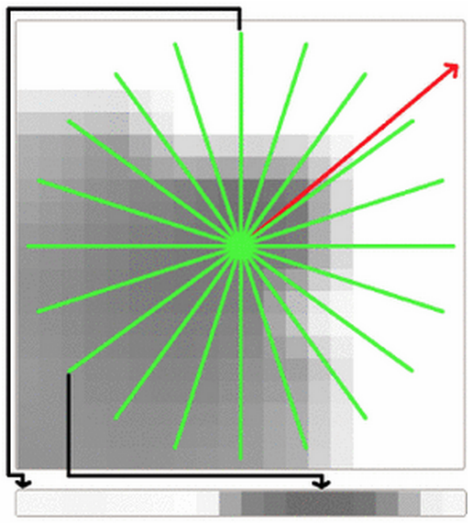


Figure 3.7 – Illustration of a NARF descriptor calculated on a range image patch. Each of the 20 green beams are represented by a single value in the descriptor vector, which is represented at the bottom. In addition, the position in the descriptor of two of those beams are marked with black arrows. The value attributed to a beam represent how pixels change under it. More precisely, beams that lie on a flat surface have low values and beams going over a high gradient (i.e. potential object border) have high values. The red arrow (pointing to the top right) show the extracted dominant orientation around the normal of the patch. From Steder et al. [2011a].

This is possible due to the fact that we know the normal to the local surface patch and the unique orientation around the latter. Figure 3.7 illustrates how descriptors are computed.

Scans Comparison

The last item to be discussed in this subsection is the method used to compare two scans. This is usually accomplished using the descriptors representing the scans. Since a descriptor is a vector of values representing the underlying data, two descriptors with very similar corresponding values should also represent similar entities (e.g. objects). Consequently, similarity between descriptors are generally computed using a simple distance metric (e.g. Euclidean distance, cosine distance).

The comparison between two scans using global descriptors is usually computationally inexpensive. Indeed, a single descriptor is computed for each scan and the comparison between two of them simply requires the computation of only one distance metric. Also note that for global descriptors, there is no such thing as keypoints and the geometric information is intrinsically included in the descriptor itself. A major drawback of such comparison approach is the high sensitivity to local changes of the environment, which may for example be caused by dynamic objects. This approach is also sensitive to small changes in position of the sensor.

From a computational point of view, scans comparison based on local descriptors are more demanding. Indeed, besides having to calculate several keypoints and their associated descriptors, a method must

be chosen to compare two scans over all features. In addition, the geometric relationship between features provides important cues about scans and may thus be considered to improve reliability of place recognition, albeit at a greater processing cost. Fortunately, using local features generally yield more robust results regarding local changes and also provides more comparison flexibility for the user. We will see a number of techniques for comparing scan using local features in the following paragraphs.

A first approach to compare *3D* scans with local descriptors consists in finding local descriptors correspondences between the scans. These correspondences are used to check if there is a valid transformation that aligns the scans. As we deal with data from real, noisy sensors, the vectors of real numbers representing the features (i.e. the descriptors) will never be exactly the same from one scan to another. A simple solution to this problem is to use the concept of nearest neighbour to identify the correspondences. One has to bear in mind that several descriptors may have no valid match due to background clutter or change in the view point. [Lowe, 2004, Section 7.1] describes how to remove most of those false matches by comparing the distance of the closest neighbour to that of the second-closest neighbour. The intuition is that this second-best match is an incorrect one and using this ratio, only matches that have the closest neighbour significantly closer than the closest next match will be used, therefore improving reliability.

Once the corresponding descriptors between two scans have been identified, they are used to determine if there is a valid rigid *3D* transformation that aligns the underlying keypoints. The existence of such valid transformation serves as a first check on the similarity between two scans. This step also requires a criterion on the number (or ratio) of features correctly aligned, thereby identifying the scans as originating from the same place or not. This is generally achieved using the RANSAC algorithm [Fischler and Bolles, 1981]. This rigid transformation estimation is computationally expensive, but has the advantage of being very discriminative. Additionally, it provides relative pose between scans, which is not possible using global descriptors. The relative pose can, for example, be used to determine odometry or during map creation process. Figure 3.8 shows keypoints from two scans of our dataset, as well as examples of correspondences.

A second solution, that speed up comparison when searching for potential matches between scans, is to represent the set of features of each scan by a *BoW* [Salton and Michael, 1983]. The concept of *BoW* was first used for documents classification. In this context, *BoW* represented a document by a vector of occurrence counts of a vocabulary without taking into account their ordering. In our case, the descriptors are made up of real numbers which allows for an infinity of them; therefore they cannot be used directly as words. The solution to this problem is to use a clustering algorithm such as k-means [MacQueen, 1967] to create groups that will represent words, a process known as *quantization*. For instance, Cummins and Newman [2008] present examples of visual words that typically correspond to the cross-piece of windows and other words that correspond to top-left corner of windows. An advantage of using k-means is that it is an unsupervised algorithm, meaning that no manual labelling effort is required. Because the *BoW* approach avoids having to compare all potential

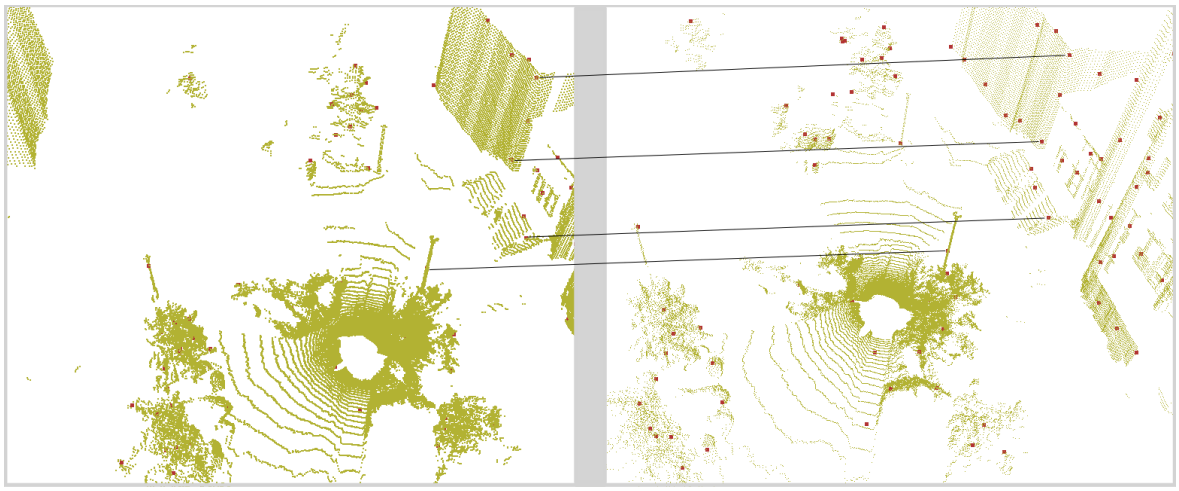


Figure 3.8 – Examples of NARF keypoints (red squares) found for two different scans of the structured dataset. Black lines illustrate examples of valid correspondences found across scans. These NARF keypoints show stability under changes, such as viewpoint, noise and resolution.

feature correspondences between the scans, it significantly reduces the processing time. On the other hand, this method removes all information regarding the geometric configuration of keypoints in the scans, which might induce unwanted perceptual aliasing [Mariottini and Roumeliotis, 2011]. *BoW* is a rather general representation for which the comparison is relatively fast to compute. Note that the visual dictionary is generally created in advance, in offline manner, using a large collection of local descriptors gathered under similar conditions and for similar scenes.

The chosen place recognition method depends on several factors such as the available hardware resources, the desired processing time and the required reliability of the results. Using features correspondences along with geometric check is more computationally expensive than using the *BoW* approach, but results are also more reliable. However, we will see in Section 3.3.2 that it is possible to take advantage of the combined use of these methods.

3.3.2 Overview of the Algorithm

In the following section, we will describe the algorithm used for our experiments. We will first describe the general idea of the technique and give a brief overview of the two articles behind it [Steder et al., 2010, 2011b]. For the place recognition algorithm to be used, one must first acquire a set of *3D* scans at approximately regular intervals with a robot equipped with an omnidirectional *LiDAR*. This initial set of scans for a given environment, dubbed scans database (D), act as a representation of all known and visited places. Subsequent scans are used to determine if a match can be found in this database, and thus establish metrics such as precision and recall. Note that this experimental procedure was repeated in different indoor and structured outdoor environments and with different *LiDAR*s for both original articles.

In the first version of the algorithm [Steder et al., 2010], they compared a newly acquired scan (z^*)

Parameters	Values for *BoW*	Values for the transformations scoring
Max. feature counts	2000	200
Descriptor size	36	36
Support size	1/10 avg. range	1/5 avg. range

Table 3.4 – The set of NARF parameters used for the *BoW* pre-ordering step and the candidate transformations scoring step. The support size is the radius around the keypoint, in the *3D* space, used to compute the descriptor. Note that the support size is represented by a proportion of the average range of all points in the database.

against all scans from the database $(z^1,...,z^{|D|})$, only using NARF features correspondences. As mentioned previously, these features encode a full *3D* pose, which in theory allow the determination of the transformation that aligns two scans based on a single features correspondence. This meant that there was as many candidate transformations as there were features correspondences between a newly acquired scan (z^*) and a single scan (z^i) from the database. Using a series of rules, a score was assigned to each of these transformations. This score reflected the system belief that this transformation was an actual match. The transformation having obtained the highest score was used to align z^* with z^i.

Although this technique yielded high recognition rates, it required to compute a score for all candidate transformations (i.e. features correspondences) between z^* and all the scans in the database $(z^1,...,z^{|D|})$. For real-time applications, this process is therefore too computationally expensive. An enhanced version of the algorithm, that we used for our experiments, was presented in Steder et al. [2011b]. In this version, they first computed a *BoW* representation for all scans. Based on this representation, they ordered all scans of the database from to the most similar (i.e. smallest Euclidean distance) to the least similar, relative to z^*. The calculation of scores for candidate transformations was then performed following this order. Consequently, the algorithm could be stopped when no more time was available and only matches that are less likely valid could be ignored.

Algorithm 1 presents a high level pseudo code of the general scan matching process. A first general note is that two sets of NARF features are extracted for each scan. This is because there is two general steps based on the features: the pre-ordering of the scans from the database using the *BoW* representation and the scoring of candidate transformations based on features correspondences. The section based on *BoW* is presented from line 1 to 12 and uses a given set of feature parameters, while the second section for scoring candidate transformations is presented from line 14 to line 36 and use a different set of parameters. Table 3.4 shows the feature parameters used for those two cases. Authors explain that: "For the BoW approach a high number of features describing small parts of the environment is most useful... However, when matching a new query [scan] z^* against [the database] D, a smaller number of more distinctive features is needed".

The first processing by the algorithm is the creation of the descriptors for all scans in the database (line 1), which are then used to create the dictionary (line 2). Note that, it would be possible to create

Algorithm 1 High Level Place Recognition Process [Steder et al., 2011b]

Input:

scanDatabase : the complete database of previous scans $(z^1, ..., z^{|D|})$

z^* : the new scan to be matched and localized against the *scanDatabase*

Output:

potentialMatches : set of (*scanDatabase* potential match for the z^*, best relative transformation, transformation score)

```
1:  allBowDescriptors ← CALCULATEALLDESCRIPTORSFORBAGOFWORDS(scanDatabase)
2:  dictionary ← CREATEDICTIONARYFORBAGOFWORDS(allBowDescriptors)
3:
4:  function FINDPOTENTIALMATCHES(scanDatabase, z*, dictionary)
5:      newScanBow ← COMPUTEBAGOFWORDS(z*, dictionary)          ▷ Keep scan reference
6:
7:      initSimilarities ← ∅                ▷ Set of (scan reference, initial similarity score)
8:      for all scan ∈ scanDatabase do
9:          scanBow ← COMPUTEBAGOFWORDS(scan, dictionary)
10:         initSimilarities.append(GETINITSIMILARITY(scanBow, newScanBow))
11:     end for
12:     sortedSimilarities ← SORTBYSCORE(initSimilarities)
13:
14:     potentialMatches ← ∅, i ← 1
15:     newScanFeatures ← GETSCANFEATURES(z*)          ▷ Each feature encodes the 3D pose
16:     while time ≤ timeout and i ≤ |scanDatabase| do
17:         scan ← GETSCAN(sortedSimilarities_i)
18:         scanFeatures ← GETSCANFEATURES(scan)          ▷ Each feature encodes the 3D pose
19:         sortedMatches ← GETSORTEDFEATUREMATCHES(scanFeatures, newScanFeatures)
20:
21:         j ← 1, bestTransfoScore = −∞, bestTransfo ← null
22:         while j ≤ |sortedMatches| and j ≤ 2000 do
23:             transfo, score ← COMPUTEMATCHTRANSFOANDSCORE(sortedMatches_j)
24:             if score ≥ scoreAcceptanceThreshold and score ≥ bestTransfoScore then
25:                 bestTransfoScore ← score
26:                 bestTransfo ← transfo
27:             end if
28:             j ← j + 1
29:         end while
30:
31:         if bestTransfo ≠ null then
32:             potentialMatches.append(scan, bestTransfo, bestTransfoScore)
33:         end if
34:
35:         i ← i + 1
36:     end while
37:
38:     return potentialMatches
39: end function
```

a dictionary using a different set of scans (i.e. not the scans from the database) and/or update this dictionary according to some criteria, but this is not the case here. Once the descriptors have been produced, they are used to create 200 words (i.e clusters) using k-means. The output dictionary is then used to create a *BoW* representation of each scan (line 5 and 9). Based on this representation, all scans from the database are stored in ascending order, according to the Euclidean distance between their vectors representing word counts and the vector of the input scan (line 10). The intuition is that scans with a small distance between them are more likely to originate from the same place. They will therefore be processed first during the next step, as they are the most promising. This greedy approach is required because of the timeout (line 16) that might prevent the last scans to be processed.

The features created using the set of parameters for candidate transformation scoring are used to find potential transformations between each database scan and the input scan (z^*). As indicated previously, each NARF feature encodes its full *3D* pose, therefore a single features match allows the determination of the transformation that should align the two scans. Since there is 2000 features per scan, there are up to 2000 transformations to be tested for each scan from the database. To determine the corresponding features, all pairs of features (one from the processed database scan and one from the input scan) are ordered by ascending order according to their Manhattan (i.e. L1) distance in the space of descriptors. Again, descriptors with a small distance between them are more similar and therefore more likely to be valid matches; they are therefore prioritized for the scoring process.

Although the candidate transformation scoring process is not detailed (line 23), it is an important part of the place recognition framework that we will briefly describe here. As explained in Section 3.3.1, range images are linked to the *LiDAR* sensor model. Each pixel represents the range from the origin of the sensor to the target for a specific angular position. Considering that the step of aligning the input scan (z^*) with the database scan (z^i) is already completed, one can easily determine where the pixel p^* from z^* will fall into z^i, as well as the range value it should have. Let r^* be the range of the processed point in z^* and r^i the range of the corresponding point in z^i. A score representing how good r^* is explained by r^i can be computed according to different scenarios presented in Table 3.5.

This scoring process is applied to all validation points and the final score for the candidate transformation is a function (see [Steder et al., 2011b, Equation 1]) of those individual point scores. To avoid a small error in the transformation to cause low score, they not only consider the exact matching points, but also neighbours in small pixel radius.

Finally, note that the set of validation point is a subsample of points, since using all points from the input scan would be too expensive to compute. Instead, the set of points used is fixed to a predefined size (200 points for the experiments). Points are chosen to evenly cover the *3D* space and to have some significance in the scene.

This concludes the general overview of the place recognition algorithm [Steder et al., 2011b]. In the following section, we will explain how we used it to compare the impact of different environments on the recognition performance.

Ranges relation	Interpretation	Effect on the score
$\lvert r^i - r^* \rvert < \Delta r_{max}$	The difference between r^i and r^* is within the confidence range. This is most likely a valid correspondence.	Increase $\propto 1 - \frac{\lvert r^i - r^* \rvert}{\Delta r_{max}}$
$r^i - r^* > \Delta r_{max}$	The range of the pixel from the database scan is larger than the corresponding pixel from the input scan. This could be caused by a dynamic or partially transparent obstacle, but this is more likely caused by a wrong transformation.	Highly decrease
$r^i - r^* < -\Delta r_{max}$	The range of the pixel from the database scan is smaller than the range of the corresponding pixel from the input scan, leading to two subcases :	
	1) A known obstacle in the database scan hides the pixel from the input scan.	Slightly decrease
	2) The pixel does not exist in the input scan. This could be caused by an unseen or dynamic obstacle, but it is more likely caused by a wrong transformation.	Highly decrease
$\nexists : r^i \mathrel{\widehat{=}} r^*$	The pixel from the input scan ends up outside of the database scan limit (i.e. there is no reference for this point).	Slightly decrease
$r^i \geq MAX$	The range of the pixel from the database scan is larger or equal to the sensor maximum reading range, leading to two subcases :	
	1) Based on the value of the input scan pixel, the corresponding pixel of the database should be closer to the sensor.	Highly decrease
	2) The pixel of the database scan moved away from the sensor and could possibly be out of range.	Moderately decrease

Table 3.5 – A summary of the different scenarios considered for scoring corresponding pixels of the range images. The range of the pixel from the input scan is r^* and the range of the corresponding pixel from the database scan is r^i. Note that, Δr_{max} is a positive value representing the maximum difference in range between the pixels to be considered as a valid correspondence and MAX is the maximum range for the given sensor.

3.4 Results

In this section, we use the algorithm previously described [Steder et al., 2011b] to evaluate place recognition performance. We first explain how we produce the data for this evaluation and discuss about some general observations about the results for our three datasets. We then analyze the results between the unstructured and structured environments as well as between the SICK and the Velodyne acquisitions.

3.4.1 Data and General Observations

In order to evaluate the place recognition performance, two elements are required. Firstly, a rule for labelling two positions of the real world as belonging—or not—to the same place, and secondly, the algorithm prediction for two input scans. In the following subsection, we first discuss these two elements and present the resulting data. Afterwards, we explain some observations about that data.

Real-World Places

As seen in previous work on topological mapping [Valgren and Lilienthal, 2008, Brunskill et al., 2007] the notion of a place is ill-defined. A simple rule can be established, however: the closer two points of the space are to each other, the more likely they are to be considered in the same place. Therefore, we use the real world physical distance between two acquisitions (d) as an indicator of the likelihood that they represent the same place. We will now discuss the method used to determine the distance between scans.

We first use the robot wheel odometry as a rough approximation of the relative pose between two consecutive scans. In order to reduce the pose estimation error, we then use the *Iterative Closest Point (ICP)* algorithm to align their respective point clouds and adjust the odometry accordingly. These steps are performed sequentially for all scans of the first loop. This process cannot be repeated independently for the second loop, because the drift accumulation would differ from loop to loop. The result would be that the calculated physical distance between a point in the first loop and its (true) corresponding point in the second loop might be significantly erroneous. To address this problem, the *ICP* odometry adjustment of the second loop is performed relative to the first loop, thus ensuring that this difference in drift do not happen. To do this, we first align the first scan of the second loop relative to the first scan of the first loop using *ICP*. Thereafter, each scan of the second loop is adjusted with respect to the scan of the first loop, that theoretically is being the closest, considering the last corrected pose and the movement of the robot. Figure 3.9 illustrates the resulting pair of paths for each of our three datasets. Note that this is a *2D* representation for which the vertical component is ignored. This figure shows the absence of discrepancy between the two loops of each dataset.

With the corrected odometry, it is possible to easily obtain the distance between two scans. To do this, one can simply calculate the norm of the difference in position between the two scans. On the other hand, it is important to remember that there is an accumulation of drift in the computed odometry,

making the final loop slightly deformed (visible in Figure 3.9 (b) (c)). This results in a significant error on the estimated distance between two scans ($scan_i$ and $scan_j$, where $i < j$) that are separated by several acquisitions; for instance, between the first scan and the last scan. To address this problem we consider a new path going from $scan_j$ to $scan_i$, in addition to the previously computed path going directly from $scan_i$ to $scan_j$. This new path must pass through the loop closure portion (i.e. the link between the last scan and the first scan). Again, we used *ICP* to calculate this portion of the path. Finally, the path for which the scans are closest in terms of acquisition numbers, from $scan_i$ to $scan_j$ or from $scan_j$ to $scan_i$, is used for the final calculation of the distance.

Note that all alignment is visually inspected to ensure that *ICP* had indeed converged to a valid solution. When this is not the case, the odometry provided by the robot is manually adjusted to enable this convergence. The second row of Figure 3.10 contains the resulting distance matrices for all pairs of scans of each of our three datasets. Note that since a distance function is by definition symmetrical, the distance matrices are symmetric by definition. The values on the main diagonal represent the distance between a scan and itself and are therefore null. A secondary diagonal of low values is produced by the small distances between the scans (d) of the first and the second loop. Finally, because we tried to stop the loop acquisitions approximately at the starting point, we observe small values at those junction points (at the bottom left corner of the distance matrix for instance).

Algorithm Prediction

For our experiments, we use a *C++* implementation of the place recognition algorithm developed by Steder et al. [2011b], who gratefully provides us with the source code. The software produce a score between 0 and 1 for each pair of scans in the database. As explained in Section 3.3, this score reflects the system belief that two scans represent the same place. More precisely, when the score is zero, the algorithm believes there is no chance that the scans originate from the same place, and when the score is one, the algorithm is certain that they do not originate from the same place.

The second row of Figure 3.10 represents the scores matrices for our three datasets. Note that the main diagonal of these matrices are the scores of a scan compared to itself, which always results in a value of 1. It is possible to observe a slight asymmetry in the matrices caused by the non-symmetric function used for calculating scores. Note that, for our results analysis, we only consider the values from the area below the main diagonal.

General Observations

Considering the two previous subsections, we should expect to see a high score for a pair of scans close to each other and a low score for a pair of distant ones. This relationship can actually be observed by the similarity of diagonal patterns between the distances matrices and the scores matrices (first and second rows of Figure 3.10). The third row of Figure 3.10 illustrates this relation with a scatter plot, where each pair of scans is represented by a single data point relating the physical distance between

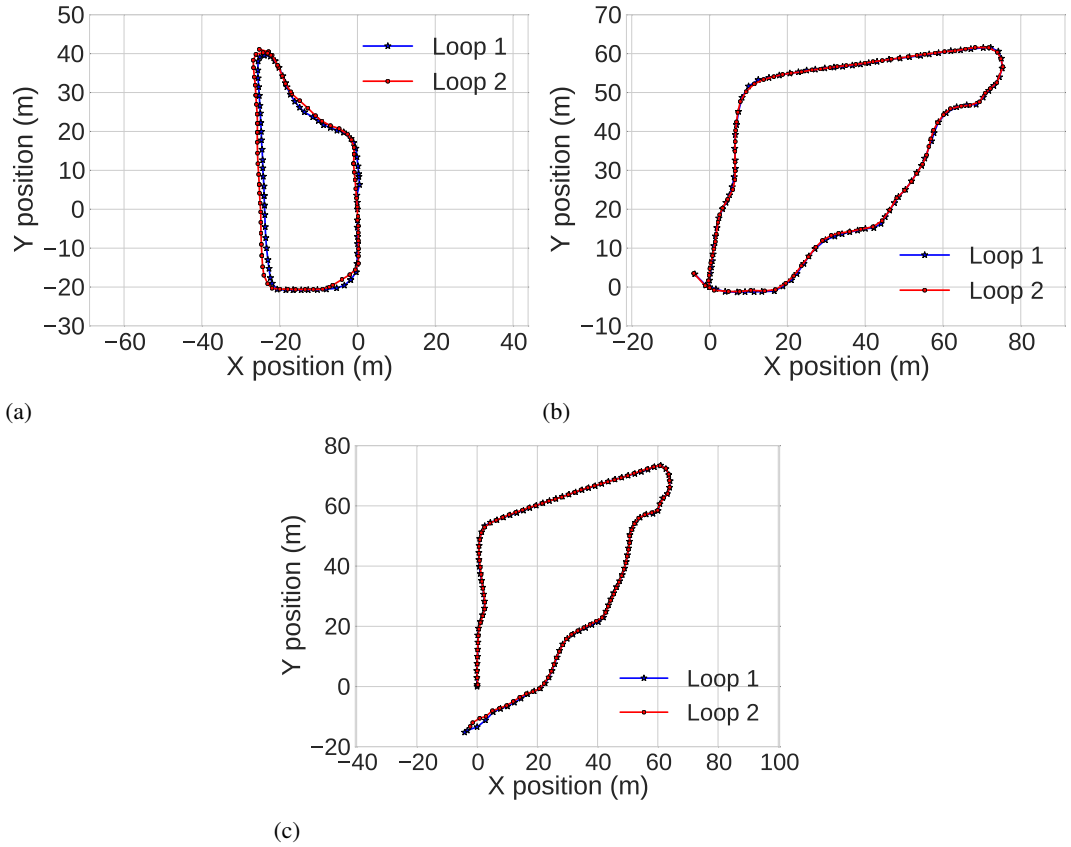


Figure 3.9 – The path resulting from the odometry adjusted with *ICP* for the two loops of each of our three datasets. (a) shows the path for the `Structured-SICK` dataset, (b) shows the path for the `Unstructured-SICK` dataset and (c) shows the path for the `Unstructured-Velodyne` dataset. Note that each marker represents the location of an acquisition.

the scans (d) and the score produced by the place recognition algorithm. The set of data points seems to follow a function of the form $f(x) = 1/x$, which also confirms our intuition.

While it is interesting to observe the complete continuous relationship between these values, practical uses of place recognition generally require a binary labelling of each pair of scans (either originating from the same place or not). To determine the ground truth labels (real-world places), a threshold $T_{distance}$ is defined on the distance between the scans (d). All pairs of scans closer than $T_{distance}$ are considered as being in the same place. Similarly for the algorithm predictions, a threshold T_{score} is used in conjunction with output score. All pair of scans obtaining a score higher than this threshold T_{score} are labelled as a match (i.e. the scans originate from the same place). Table 3.6 summarizes how these two thresholds are used to label data and analyze results. Note that $T_{distance}$, as opposed to T_{score}, $T_{distance}$ is not a parameter of the place recognition system, but rather a tool for quantifying results.

The value of $T_{distance}$ to consider during the evaluation of results depends on different factors, such as the environment in which the place recognition algorithm is used. For instance, one might consider that for indoor environments, places are generally very close from each other (e.g. closer than 4 m).

	$\mathbf{d} < \mathbf{T_{distance}}$	$\mathbf{d} >= \mathbf{T_{distance}}$
$\mathbf{s} > \mathbf{T_{score}}$	*True Positive (TP)*	*False Positive (FP)*
$\mathbf{s} <= \mathbf{T_{score}}$	*False Negative (FN)*	*True Negative (TN)*

Table 3.6 – A summary of scans pairs labelling for results analysis. d represents the distance between the two scans and s represents the place recognition algorithm output score. $T_{distance}$ and T_{score} represent the distance threshold and the score threshold, respectively.

In our context of open outdoor environments, we consider that the value of $T_{distance}$ is mainly limited by the maximum measurement range of the sensor (i.e. approximately 50 m), because over two times this measurement range, there is no possible overlap between the scans. Choosing a higher value for $T_{distance}$ implies that the robot can be further away from the previously visited place and should still be able to recognize it. Consequently, maintaining relatively high scores even for higher $T_{distance}$ values is often preferable, because it allows greater flexibility during the robot navigation. The fourth row of Figure 3.10 shows how the recall decreases as $T_{distance}$ increases, which can be explained by the reduction of overlap between scans, making the place recognition task more difficult.

As indicated in the introduction of this chapter (Section 3.1), *SLAM* algorithms often use place recognition to detect loop closures. When the robot detects that it is in a place visited before, it can connect (or merge) these concordant regions and adjust the map accordingly. If two scans are falsely identified as coming from the same place (i.e. a *FP*), the *SLAM* algorithm will distort the map while trying to connect them. In contrast, if a place is not recognized (i.e. *FN*), the algorithm simply does not benefit from this cue to adjust the map, without further adverse results. Therefore, *FP* are significantly more harmful than *FN* (sometimes catastrophic to the point of rendering the map useless) and must be avoided.

A solution to avoid *FP* is to rely on high-confidence matches (i.e. by choosing a score threshold that is high enough). In our experiment, the minimum values for this threshold are depicted by the line in the graphs of the third row of Figure 3.10. In other words, if T_{score} is above that line for the considered value of $T_{distance}$, this will result in no *FP*. Moreover, the closer the value is from that line, the better the recall will be. In our case, the recall represents the fraction of real-world places that are correctly identified by the algorithm (i.e. $TP/(TP+FN)$). The recall is presented as a function of $T_{distance}$ for different score thresholds in the fourth row of Figure 3.10. One can see that, as we pick a lower threshold value T_{score}, the recall rate increase, as we become more permissive for our matches.

3.4.2 Comparative Analysis

In the previous subsection, we explained how we produce the results and we discussed some observations for our datasets from a general perspective. We now focus on the main interest of this document, and analyze how the algorithm behaves in the unstructured environment compared to the structured environment. We also conduct this comparative analysis between two *LiDAR*s, the SICK LMS151

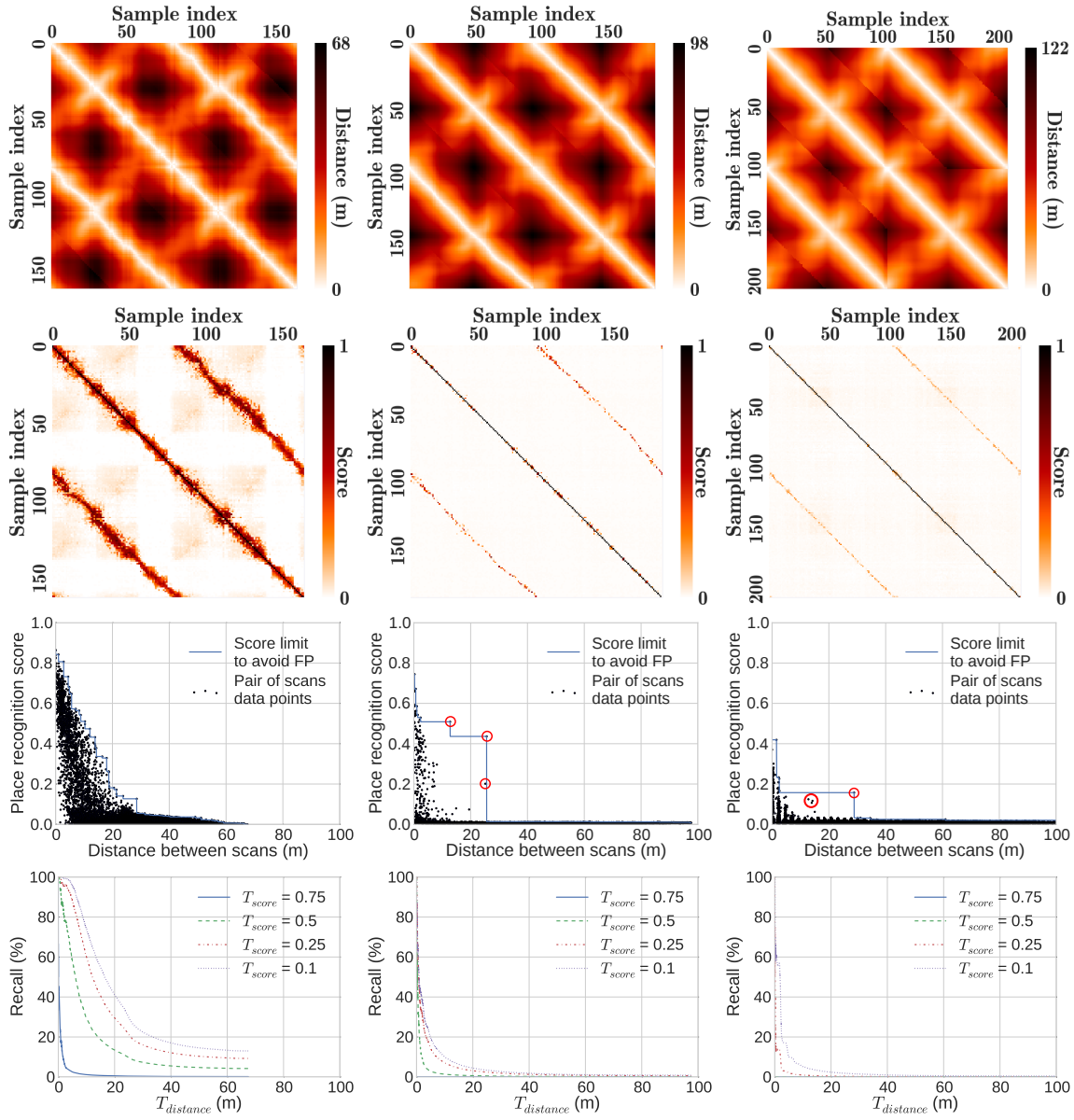


Figure 3.10 – Place recognition results overview for our three datasets. **First column:** The `Structured-SICK` dataset. **Second column:** The `Unstructured-SICK` dataset. **Third column:** The `Unstructured-Velodyne` dataset. **First and second rows:** The matrices of distances and scores, respectively. The axes show the ordered sample indices for both loops sequentially. Consequently, all pairs of scans are represented by a position in the matrices and the colour represent the distance between the scans of a pair (first row) and the place recognition algorithm output score (second row). **Third row:** The association between the distance separating the scans and the score obtained for all scans pairs, as well as the score limit to avoid any *FP* as function of the distance between the scans. Outliers are circled in red. **Fourth row:** The recall rate as function of the maximum distance (m) between scans to be considered as originating from the same place ($T_{distance}$).

and the Velodyne HDL-32E. For these comparisons, we consider how the score changes as function of d and look for outliers. Finally, even if the current experimental configuration does not allow to determine the exact source of the differences in results, we suggest some possible causes.

Structured and Unstructured Environments

For this comparison, we expect the performance of the algorithm to be worse for unstructured environments than for structured environments. Indeed, we think that the uneven ground, the significant occlusions and the presence of many small structures (e.g. branches, leaves) in unstructured environments are harmful to the *LiDAR*-based place recognition task. In order to evaluate this hypothesis, we look at the score as function of d (third row of 3.10). As noticed previously, the distribution of points seems to represent a function of the form $f(x) = 1/x$. In the case of the structured environment, no point appears to significantly challenge this distribution, but for the unstructured environment, there seems to be outliers (circled in red) when the value of d is around 13 m and 25 m. To avoid getting *FP* caused by these outliers, the score threshold must be set higher, therefore reducing the recall (see Figure 3.10 row four).

Another element to suggest that the performance is better in the structured environment is the value of d for which the score stop decreasing significantly as the value of d increase (i.e. where the distribution becomes almost horizontal). It can be observed in the third row of Figure 3.10 that this happens for d of approximately 30 m in the structured environment and approximately 15 m in the unstructured environment. For d above these values, all data points are spread almost evenly under a horizontal line (i.e. score value). It is therefore impossible to discriminate pair of scans that originate from the same place from those that do not originate from the same place based on the algorithm output score. In other words, the algorithm is able to correctly identify places in a larger radius for the structured environment than in the unstructured environment.

We will now propose a few possible causes as to why the results are better in the structured environment than in the unstructured environment. As indicated in Section 3.2, the average acquisition range is smaller for the unstructured datasets. Therefore, because of parallax, a given movement of the robot will have a greater influence on the scan content in this dataset. In addition, the unstructured environment contains many objects distributed throughout the reachable space by the *LiDAR*, which in conjunction with the parallax, creates many different occlusion patterns. In other words, the objects or parts of objects captured by the *LiDAR* greatly change as we move the robot. Another element to consider is the types of object that are present in the various environments. For instance, the objects present in the structured environment, such as buildings, are potentially better represented by the NARF features. Indeed NARF features were tested on man-made objects Steder et al. [2011a] and in structured environments Steder et al. [2010, 2011b]. Finally, because of the structured environment ground is almost entirely flat, the alignment between two scans is summed almost entirely to a translation along the XY plane and rotation about the Z axis. The additional degrees of freedom (i.e. translation along the Z axis and rotations about X and Y axis) caused by the uneven ground in

the unstructured environment are likely to make the scans alignment process of the place recognition algorithm more challenging.

SICK and Velodyne *LiDAR*s

In this section, we are interested in comparing the SICK and the Velodyne performance for place recognition, as it might help understand how important the choice of the sensor is for this particular task. In our case, we expected the SICK mounted on the *PTU* to have better results for place recognition, as it retrieves more information about the scene than the Velodyne. As a reminder, Section 3.2 and more precisely in Table 3.2, provide details about these sensors and the acquired data. One can notice that, although the Velodyne has a better horizontal resolution than the SICK, it has a smaller vertical resolution and *FOV*, as well as a smaller total point counts (refer to Table 3.2).

Our first interesting observation, in the third row of Figure 3.10, is that all the scores obtained with the Velodyne are below 0.5. This means that the algorithm has very low confidence that any pair of scans originate from the same place, even when those scans are very close physically from each other. In comparison, the data obtained with SICK produce scores above 0.7 for scans that are really close. Consequently, data points are concentrated in the lower part of the graph and the place recognition's results will be more sensitive to the choice of score threshold. We can also confirm these observations with the graphics of the fourth row of Figure 3.10. There, one can see that for all of the Velodyne data, the recall is 0 when the score threshold is set to 0.75 or 0.5. There is also a greater gap between the two other curves (i.e for score thresholds of 0.25 and 0.1) over the corresponding curves for SICK.

On the other hand, there seems to be no major differences between those distributions regarding the value of d for which the score stop decreasing significantly (around 10 m to 15 m). One could argue that this maximum range is simply due to the closeness of the environment. In addition, it does not seem to be any significant difference in the amount or distribution of outliers, which is important to avoid *FP*.

It can be concluded that there is indeed a difference between the results obtained for these two sensors. However, this difference may be insufficient to counteract other practical factors. For instance, one could choose to use the Velodyne for its higher data throughput, despite its slightly worse performance, and thus avoid having to immobilize the robot for each acquisition.

3.5 Conclusion

This chapter focused on the impact of the level of structure (man-made vs. natural) of an environment on the performance of a *LiDAR*-based place recognition algorithm. As explained, these algorithms are key to solving several important problems of mobile robotics, such as loop closures detection for *SLAM*. Our research is therefore useful to assess whether an existing solution based on NARF, currently operating in structured environments, can also be used successfully for more complex, unstructured environments. Our experiments were based on real datasets collected with a Husky A200

robotic platform, using either the SICK or the Velodyne *LiDAR* as the main sensing device in the two desired types of environments. An outdoor area containing several man-made structures, including buildings, was used as our structured environment and a forest area was used as our unstructured environment. Paths forming a loop were followed by the robot twice in each environment to produce examples of recognizable places.

After summarizing the concepts of feature keypoints and descriptors, we explained how to convert point clouds into range images and use this data representation to extract a specific type of feature called NARF. As mentioned, these features have the advantage to consider only edges produced by objects boundaries, as opposed to edges caused by occlusion, making them a sensible choice for place recognition in complex environments. Thereafter, we described the place recognition algorithm we choose for our comparative analysis, which was presented in [Steder et al., 2011b]. This algorithm uses NARF features with a technique combining *BoW* and features matching for scoring two input scans, according on how they are likely to originate from the same place.

The physical distance between two scans was used to determine whether or not they are considered to be in the same place. This value was obtained from the odometry corrected using *ICP*. We expected to see a generally decreasing relationship between the score of the place recognition algorithm and the distance between the input scans, which turn out to be true for all our datasets. The results were also presented using discrete labels to be more faithful to practical uses, in which two scans either originate or not from the same place. Using these labels, it was possible to identify *FP*, which must be avoided when the place recognition algorithm is used for loop closures detection. Furthermore, this discretization allowed us, for different score thresholds, to compute the recall as function of $T_{distance}$ (i.e. maximum distance between two scans to be considered as originating from the same place).

The first important finding of our comparative analysis was that the unstructured dataset produced some outliers regarding the relation between the place recognition range and the algorithm output score, which was not the case for the structured dataset. These outliers greatly reduce the algorithm's performance when *FP* have to be avoided.

Another important observation was that the scores drop significantly faster with distance (separating the scans) in the unstructured environment when compared to the structured environment. Specifically, reliable recognition of places was only possible in a radius of less than 15 m in forest, which is approximately 15 m less than for the open environment between buildings. We also compared the results for the SICK and Velodyne *LiDAR*s in the unstructured environment. Results for those sensors were relatively similar, but the samples obtained with the Velodyne got lower scores in general. This implies that a small change of the score threshold greater impact results.

Conclusion

The objective of this document was to analyze the influence of complex environments on different *LiDAR*s used in a context of mobile robotics navigation. Because *LiDAR*s can effectively retrieve the three-dimensional structure of an environment, they can advantageously be used for several navigation tasks, such as obstacles avoidance, localization and mapping. The information they provide is complementary to that of other sensors such as cameras, which capture information on the appearance of the surrounding scene. Although the literature on mobile robotic navigation is rich, most existing research deal with indoor or structured outdoor environment. Dealing with complex environments, such as forest or falling snow conditions, remains a important problem. Therefore, analyzing how such complex environments influence the *LiDAR*-based navigation can help to develop algorithms that are more robust to changing conditions.

In Chapter 2, we focused our attention on the impact of falling snow on *LiDAR* data. For that matter, we first presented an overview of *LiDAR*s functioning, which helped understand what could potentially cause noise in the data. The first issue raised was that, when the spot produced by the laser beam is not completely on a single target, a method of inference of the distance must be chosen and may not be suitable for the application. This is generally caused by object edges of small particles. The second issue raised was that *LiDAR*s acquire data points sequentially, which can lead to a distorted representation of the scene when the environment is dynamic. Consequently, the dynamic nature and small size of snowflakes make snowfall a perfect example of challenging condition for *LiDAR* acquisition.

To perform our analysis of the impact of falling snow on *LiDAR* acquisition, we collected data from four sensors simultaneously, namely the Velodyne HDL-32E, the SICK LMS151, the SICK LMS200 and the Hokuyo UTM-30LX-EW. We acquired data during six snowfalls in a wide variety of conditions. The final dataset used for our analysis contains more than 40 hours of data.

For our analysis, we first observed how the proportion of *LiDAR* returns caused by snowflakes evolved over time. We visually represented the short-term evolution by overlaying the snowflakes according to their spatial arrangement for four subsequent scans. This showed significant quantitative and spatial changes, even over such short time. We observed no pattern in the distribution of snowflakes and believe it can be modelled by a random process. We also illustrated the long-term evolution by showing the proportion of return as function of time, for the whole duration an acquisition. The shape

of this curve provides information on the progress of the weather during this period. Additionally, we conducted a comparative analysis of the overall sensitivity to snowflakes of our four sensors. We showed that the SICK LMS200 is the more sensitive with peaks reaching up to 15 % of echoes caused by falling snow, while other three *LiDAR*s never exceeded 1 %. Beside our temporal analysis, we analyzed how range can affect the probability to trigger a measurement. Based on a histogram, we found that the probability density function is close to a log-normal. One final important finding is that beyond 10 m, snowflakes no longer seem to trigger measurement.

In Chapter 3, we have conducted our analysis at a higher level, by comparing the performance of a state-of-the-art *LiDAR*-based place recognition algorithm in different environments. There are two main reasons for this choice of algorithms . Firstly, the ability of a robot to identify previously visited places allows to solve several other navigation problems, such as multi-session-mapping, kidnapped robot and loop closure in *SLAM*. Secondly, the chosen algorithm proved to be successful in indoor and structured outdoor environments, but had not been tested in unstructured environments.

For our comparative analysis, we acquired datasets in two different areas of the Laval University campus using the Husky A200 mobile robot. The first area is our model for a structured outdoor environment, similar to those on which the original algorithm was tested. In this location, the ground is mostly flat and the scanned space contains several man-made objects, such as buildings, stairs and lampposts. The second area, representing our unstructured environment, is in a forest where the ground is uneven in some parts. We also used two different sensors for our acquisitions, namely the SICK LMS151 and the Velodyne HDL-32E. We used the resulting `Structured-SICK`, `Unstructured-SICK` and `Unstructured-Velodyne` datasets to evaluate the impact of both the environment and the sensor choice on the place recognition performance.

We showed the relation between the algorithm output score and the distance separating the input scans. We consider that the closer two scans are from each other, the more likely they are to represent the same place. Therefore, we expected an inverse relation between the score and the distance separating the scans. We also presented the results using binary labels, because practical uses of place recognition generally required pairs of scans to be identified as originating from the same place or not. For that matter, all scans for which the distance between them was below a threshold were considered as belonging to the same place. Similarly, we considered that if the score obtained for a pair of scans was above a threshold, the algorithm labelled the input scans as representing the same place. This labelling also enabled us to identify false positives and calculate the recall for different thresholds combinations. As we explained, false positives must be avoided during *SLAM*, because they create false loop closures, which distort the resulting map in a castastrophic manner.

The presentation of results, as described above, enabled us to make our final observations. As expected, we showed a relation of the form $f(x) = 1/x$ between the algorithm output score and the distance between two scans. This was true for our three datasets. However, some outliers were observed for the unstructured environment, whereas this was not the case for the structured environment.

This is important when using the discretized labels, because such outliers can result in *FP* or, if the score threshold is adjusted to avoid them, significantly reduce the recall. Our last observation is that the score decreases more rapidly as a function of the distance between scans in the unstructured environment than in the structured environment. This means that the algorithm becomes sensitive to disturbances and noise as the place recognition range increases more quickly in an unstructured environment than in a structured environment. The same observation applies when comparing the sensors, in which case the algorithm become sensitive faster as the range increases for the Velodyne than the SICK.

To conclude, the goal of this document was to evaluate the influence of complex environments on *LiDAR*-based robot navigation. By characterizing *LiDAR*s data acquired during snowfalls and by comparing performances of *LiDAR*-based place recognition in forest and in structured outdoor environment, we showed that these environments can negatively impact robot navigation. Based on our observations, future research could focus on finding methods to increase robustness to these conditions, for instance by filtering input data. In the case of our place recognition analysis, a better identification and quantification of the sources causing performance decrease could also help the development of better solutions for navigation.

Bibliography

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics (T-RO)*, 23(1):34–46, 2007.

Stefan Kohlbrecher, Oskar von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 155–160, Kyoto, 2011. IEEE.

Marwan Hussein, Matthew Renner, and Karl Iagnemma. Global localization of autonomous robots in forest environments. *Photogrammetric Engineering & Remote Sensing*, 81(11):839–846, 2015.

Kai M. Wurm, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1217–1222, St. Louis, oct 2009. Ieee.

Markus-Christian Amann, Thierry Bosch, Marc Lescure, Risto Myllyla, and Marc Rioux. Laser ranging: a critical review of usual techniques for distance measurement. *Optical Engineering*, 40 (1):10–19, 2001.

Thierry Peynot, Steve Scheding, and Sami Terho. The marulan data sets:multi-sensor perception in natural environment with challenging conditions. *The International Journal of Robotics Research (IJRR)*, 29(13):1602–1607, nov 2010.

Stewart Moorehead, Reid Simmons, Dimitrios Apostolopoulos, and William L. Whittaker. Autonomous navigation field results of a planetary analog robot in antarctica. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, June 1999.

Brian Yamauchi. Fusing ultra-wideband radar and lidar for small ugv navigation in all-weather conditions. In *SPIE Defense, Security, and Sensing*, pages 76920O–76920O. International Society for Optics and Photonics, 2010.

Yasushi Sumi, Byeong Koo Kim, Yuki Matsumoto, Eiichi Horiuchi, Osamu Matsumoto, and Kohtaro Ohba. Outdoor environment simulators for vision-based safety sensors; artificial sunlight lampheads and simulated-snow chamber. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 125–130, Nov 2013.

Peter C Barnum, Srinivasa Narasimhan, and Takeo Kanade. Analysis of rain and snow in frequency space. *International Journal of Computer Vision*, 86(2-3):256–274, 2010.

Henri Servomaa, Ken-ichiro Muramoto, and Toru Shiina. Snowfall characteristics observed by weather radars, an optical lidar and a video camera. *IEICE Transactions on Information and Systems*, 85(8):1314–1324, 2002.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. 2006.

Silvio Filipe and Luis A Alexandre. A comparative evaluation of 3D keypoint detectors in a RGB-D object dataset. In *International Conference on Computer Vision Theory and Applications*, pages 145–148, Lisbon, 2014.

Edmond Boyer, Alexander M Bronstein, Michael M Bronstein, Benjamin Bustos, Tal Darom, Radu Horaud, Ingrid Hotz, Yosi Keller, Johannes Keustermans, Artiom Kovnatsky, et al. SHREC 2011: robust feature detection and description benchmark. *arXiv preprint arXiv:1102.4258*, 2011.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, pages 147–151, 1988.

Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3d registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217. IEEE, 2009.

Zhong Yu. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 689–696. IEEE, 2009.

Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision (ECCV)*, pages 356–369. Springer, 2010.

Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard. Point feature extraction on 3d range scans taking into account object boundaries. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2601–2608, Shanghai, 2011a. IEEE.

Antonio Torralba, Kevin P Murphy, William T Freeman, and Mark A Rubin. Context-based vision system for place and object recognition. In *IEEE International Conference on Computer Vision*, volume 1, pages 273–280. IEEE, 2003.

Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In IEEE, editor, *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1023–1029, 2000.

Mark Cummins and Paul Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research (IJRR)*, 27(6):647–665, 2008.

Mark Cummins and Paul Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research (IJRR)*, 30(9):1100–1123, 2011.

Martin Magnusson, Henrik Andreasson, Andreas Nüchter, and Achim J. Lilienthal. Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform. *Journal of Field Robotics*, 26(11–12):892–914, 2009.

Timo Röhling, Jennifer Mack, and Dirk Schulz. A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 736–741. IEEE, 2015.

Bastian Steder, Michael Ruhnke, Slawomir Grzonka, and Burgard Wolfram. Place recognition in 3d scans using a combination of bag of words and point feature based relative pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1249–1255, San Francisco, 2011b. IEEE.

Bastian Steder, Giorgio Grisetti, and Wolfram Burgard. Robust place recognition for 3d range data based on point features. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1400–1405. IEEE, 2010.

Maxime Latulippe, Alexandre Drouin, Philippe Giguère, and François Laviolette. Accelerated robust point cloud registration in natural environments through positive and unlabeled learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2480–2487, 2013.

Jean-François Lalonde, Nicolas Vandapel, Daniel F. Huber, and Martial Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839–861, 2006.

Matthew W Mcdaniel, Takayuki Nishihata, Christopher A Brooks, Phil Salesses, and Karl Iagnemma. Terrain classification and identification of tree stems using ground-based lidar. *Journal of Field Robotics*, 29(6):891–910, 2012.

Meng Song, Fengchi Sun, and Karl Iagnemma. Natural feature based localization in forested environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3384–3390, 2012a.

Mikko Miettinen, Matti Ohman, Arto Visala, and Pekka Forsman. Simultaneous localization and mapping for forest harvesters. In *IEEE International Conference on Robotics and Automation (ICRA)*, number April, pages 517–522, 2007.

SICK. SICK LMS1xx Operating Instructions, a.

Boris Jutzi and Uwe Stilla. Precise range estimation on known surfaces by analysis of full-waveform laser. *Proceedings of Photogrammetric Computer Vision*, 2006.

SICK. Lms500-21000 lite, b. URL http://www.robotsinsearch.com/products/lms500-21000-lite.

Velodyne. Velodyne HDL-32E User's Manual, a.

Hokuyo. Scanning Laser Range Finder UTM-30LX-EW Specification.

SICK. SICK LMS151 Datasheet, c.

SICK. Technical Documentation LMS200/211/221/291 Laser Measurement Systems, December 2006.

Willow Garage. Robot operating system. URL http://www.ros.org/.

Government of Canada. Canadian weather. URL http://weather.gc.ca/canada_e.html.

Wolfram Burgard, Dieter Fox, and Sebastian Thrun. *Probabilistic robotics*. MIT Press Cambridge, 2006.

Andrew Howard. Multi-robot mapping using manifold representations. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 4198–4203. IEEE, 2004.

Clearpath Robotics. Husky. URL http://www.clearpathrobotics.com/husky-ugv/.

Universität Freiburg. Freiburg campus 360 degree 3d scans. URL http://ais.informatik.uni-freiburg.de/projects/datasets/fr360.

Andreas Nüchter and Kai Lingemann. Robotic 3d scan repository. URL http://kos.informatik.uni-osnabrueck.de/3Dscans/.

Velodyne. Velodyne HDL-32E Datasheet, b. URL http://velodynelidar.com/lidar/products/manual/63-9113HDL-32Emanual_RevE_NOV2012.pdf.

César Cadena, Dorian Gálvez-López, Juan D. Tardós, and José Neira. Robust place recognition with stereo sequences. *IEEE Transactions on Robotics (T-RO)*, 28(4):871–885, 2012.

Martin a. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, 1981.

Gerard Salton and J Michael. Mcgill. *Introduction to modern information retrieval*, pages 24–51, 1983.

James MacQueen. Some methods for classification and analysis of multivariate observations. *Berkeley Symposium on Mathematical Statistics and Probability*, 1(233):281–297, 1967.

Gian Luca Mariottini and Stergios I. Roumeliotis. Active vision-based robot localization and navigation in a visual memory. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6192–6198. IEEE, 2011.

Christoffer Valgren and Achim Lilienthal. Incremental spectral clustering and seasons: Appearance-based localization in outdoor environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1856–1861. IEEE, 2008.

Emma Brunskill, Thomas Kollar, and Nicholas Roy. Topological mapping using spectral clustering and classification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3491–3496. IEEE, 2007.

Clement Mallet and Frederic Bretar. Full-waveform topographic lidar: State-of-the-art. *ISPRS Journal of photogrammetry and remote sensing*, 64(1):1–16, 2009.

James H Lever, AJ Delaney, Laura E Ray, Eric Trautmann, LA Barna, and AM Burzynski. Autonomous gpr surveys using the polar rover yeti. *Journal of Field Robotics*, 30(2):194–215, 2013.

David Mader, Patrick Westfeld, and Hans-Gerd Maas. An integrated flexible self-calibration approach for 2d laser scanning range finders applied to the Hokuyo UTM-30LX-EW. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5(June): 385–393, 2014.

Dean Pomerleau and Todd Jochem. Rapidly adapting machine vision for automated vehicle steering. *IEEE Expert: Special Issue on Intelligent System and their Applications*, 11(2):19–27, April 1996.

Chad Rockey. sicktoolbox_wrapper, October 2013. URL http://wiki.ros.org/sicktoolbox_wrapper.

Konrad Banachowicz. Lms1xx, May 2015. URL http://wiki.ros.org/LMS1xx.

Chad Rockey. urg_node, August 2014. URL http://wiki.ros.org/urg_node.

Jack O'Quin. Velodyne, September 2013. URL http://wiki.ros.org/velodyne.

Autogenerated. sensor_msgs/LaserScan Message, May 2015. URL http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html.

Autogenerated. sensor_msgs/PointCloud2 Message, May 2015. URL http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html.

Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

Meng Song, Fengchi Sun, and Karl Iagnemma. Natural landmark extraction in cluttered forested environments. In *IEEE International Conference on Robotics and Automation*, pages 4836–4843, 2012b.

Francis Colas, Srivatsa Mahesh, Francois Pomerleau, Ming Liu, and Roland Siegwart. 3d path planning and execution for search and rescue ground robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 722–727, Tokyo, nov 2013. IEEE.

Paul J Besl and Neil D McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics (FnTROB)*, 4(1):1–104, 2015.

François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.