Unsupervised Gaussian mixture models for the classification of outdoor environments using 3D terrestrial lidar data

Artur Maligo

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue par :

Artur Maligo

le 28 janvier 2016

Titre :

Unsupervised Gaussian Mixture Models for the Classification of Outdoor Environments Using 3D Terrestrial Lidar Data

---

École doctorale et discipline ou spécialité :

EDSYS : Robotique 4200046

Unité de recherche :

LAAS-CNRS

Directeur/trice(s) de Thèse :

Simon Lacroix

Jury :

Rapporteurs :
Olivier Aycard, Maître de Conférences
Paul Checchin, Maître de Conférences

Examinateurs :
Michel Devy, Directeur de Recherche
Simon Lacroix, Directeur de Recherche

Unsupervised Gaussian Mixture Models for the Classification of Outdoor Environments Using 3D Terrestrial Lidar Data

Artur Maligo

March 22, 2016

Abstract

The processing of 3D lidar point clouds enable terrestrial autonomous mobile robots to build semantic models of the outdoor environments in which they operate. Such models are interesting because they encode qualitative information, and thus provide to a robot the ability to reason at a higher level of abstraction. At the core of a semantic modelling system, lies the capacity to classify the sensor observations. We propose a two-layer classification model which strongly relies on unsupervised learning. The first, intermediary layer consists of a Gaussian mixture model. This model is determined in a training step in an unsupervised manner, and defines a set of intermediary classes which is a fine-partitioned representation of the environment. The second, final layer consists of a grouping of the intermediary classes into final classes that are interpretable in a considered target task. This grouping is determined by an expert during the training step, in a process which is supervised, yet guided by the intermediary classes. The evaluation is done for two datasets acquired with different lidars and possessing different characteristics. It is done quantitatively using one of the datasets, and qualitatively using another. The system is designed following the standard learning procedure, based on a training, a validation and a test steps. The operation follows a standard classification pipeline. The system is simple, with no requirement of pre-processing or post-processing stages.

Contents

Chapter 1

Introduction

1.1 Context

Outdoor mobile robotics aims at designing robots to be employed in fields such as agriculture, mining, inspection, monitoring, exploration, mapping and search and rescue. The environments encountered vary from urban, characterized by artificial, relatively structured elements like buildings and streets, to off-road, characterized by natural, relatively unstructured elements like vegetation and rough terrain. Mobile robots come in different forms, including aquatic, terrestrial and aerial. This thesis considers the case of terrestrial robots.

The control of robots by a human operator, or team of operators, faces obstacles inherent to mobile robotics applications: limited communication with the robot, difficult access to the site, life-threatening hazards, repetitiveness of tasks, long duration of tasks, and more [Dudek and Jenkin, 2000]. Thus, it is mostly desirable to design robots possessing *autonomy*, in order to reduce their dependence on human intervention and increase automation as much as possible. Autonomous, terrestrial mobile robots are known as *unmanned ground vehicles* (*UGVs*), or *autonomous ground vehicles* (*AGV*s).

An autonomous mobile robot must be endowed with *perception*, *decision* and *action* abilities. The robot uses perception to extract information about itself and about the environment, then it makes decisions based on this information and on its goals, then it finally acts according to the decisions made. The focus of this thesis is on perception. Important examples of perception problems are terrain traversability analysis [Papadakis, 2013], detection of zones-of-interest, detection and tracking of objects-of-interest [Nüchter and Hertzberg, 2008] and data association [Thrun, 2003].

Underlying the above-mentioned problems, and providing the base for their solution, is the more fundamental problem of *environment modelling*. Many types of environment models exist, each one containing information of a specific nature and being exploited in

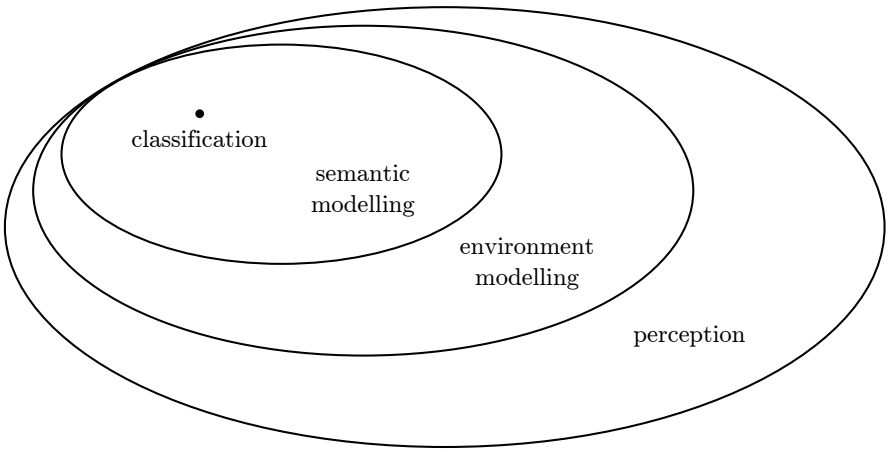


Figure 1.1: *Perception Problems.* A view to perception problems through sets. Classification is a stage of semantic modelling, therefore the classification problem is an element of the set of semantic modelling problems. Any semantic modelling problem, in turn, is also an environment modelling problem, and thus semantic modelling problems constitute a subset of the environment modelling problems. The same follows for environment modelling and perception.

a different manner by other processes of a robot. *Semantic models* [Rusu, 2009], in this context, are especially interesting because they encode qualitative information, and thus provide to a robot the ability to reason at a higher level of abstraction. More specifically, we are interested in *classification*, which is one of the core components of a semantic modelling system. The relations just explained are illustrated in figure 1.1.

To perceive the outside world, robots use sensors such as cameras, radars and lidars. Whereas cameras provide information, in colours or in greyscale, about the visual texture and shape of elements, radars and lidars provide information about the geometry of elements, and in some cases about their reflectivity. Lidars, compared to radars, provide more reliable and accurate measurements, which allows the use of simpler processing methods [Adams et al., 2011]. In this work, we focus on lidars. The sensors' observations are represented as *3D point clouds*, which constitute the input of our system and allow the robot to perceive and interpret the environment based on its geometry.

1.2 Problem Statement

The problem of classification can be generically defined as the assignment of labels to the classification elements or inputs. The labels possess a semantic interpretation, and thus the resulting model is semantic, providing qualitative information that can be exploited

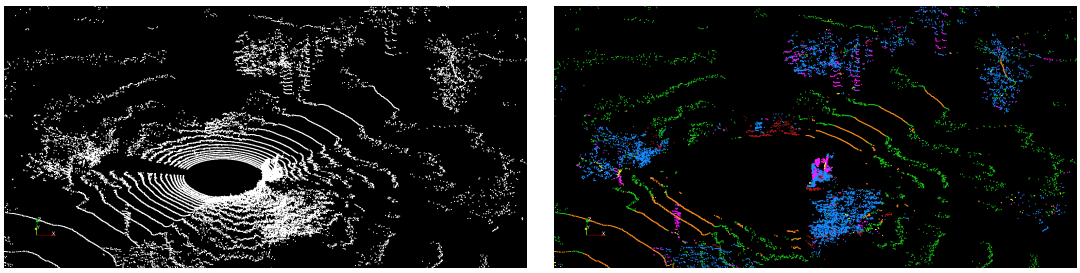


Figure 1.2: *Classification Example.* The left image shows the input point cloud, and the righ image shows the output of the classification. The lidar was onboard a UGV, which was located in the middle of the empty circle at the center of the image. Colours encode the class labels: (*road*, orange), (*trunk*, pink), (*vegetation*, blue), (*grass*, green), (*rough*, brown). From the right image, it is possible to see that this outdoor scene contains a road, surrounded by some grass, shrubs and trees. A person, the robot operator in this case, can be seen standing on the road, at the left. It is clear that points of the input cloud are missing in the output. The classification of these points was ambiguous, and thus the system left them out of the final result, marking them as *unknown*.

in high-level tasks. Figure 1.2 shows an example where classification is applied pointwise, considering an input 3D point cloud acquired with a lidar.

The essencial information encoded in a point cloud is the geometry of the elements in the world. However, interpreting the world's geometry through a point cloud is not a simple task. Difficulties arise, firstly, from the variability encountered in outdoor environments, which contain elements of all shapes and scales, possibly cluttered together. Secondly, difficulties arise from the manner in which scene elements are sampled by a lidar, which depends on their position relatively to the sensor, on occlusions, and on the sampling pattern of the lidar.

We propose to approach the classification problem using the framework of *probabilistic classification* [Bishop, 2006]. Although supervised learning can be employed, it is not scalable with respect to the amount and complexity of the concerned data, due to the necessity of manual labelling by a human domain expert. An alternative is to apply unsupervised learning, which overcomes this necessity by automatically discovering the classes that are represented in the data. The challenge becomes then to ensure that these classes can be semantically interpreted, and thus exploited in some task.

1.3 Contributions

Our main contribution, forming the core of our proposed approach, is a two-layer classification model which mainly relies on unsupervised learning. The first, intermediary layer consists of a Gaussian mixture model. This model is determined in the training step in an unsupervised manner, and defines a set of intermediary classes which is a fine-partitioned representation of the environment. The second, final layer consists of a grouping of the intermediary classes into final classes that are interpretable in the considered target task. This grouping is determined by an expert during the training step, in a process which is supervised, yet guided by the intermediary classes.

The intermediary classes are used to represent the environment variability and the different sampling conditions encountered in the data. The intermediary layer is thus data-oriented, serving as an abstraction for the data factors that influence the classification. The final classes, in turn, can be semantically interpreted as useful entities for the target task. The final layer is therefore task-oriented, introducing the task-dependent factors in the classification. The classification model, as a whole, is a predictive model and can be used to classify new data.

A normal, full application of the approach consists in: (a) data acquisition; (b) composition of the learning datasets; (c) feature extraction, unsupervised training and supervised grouping for a few different systems to be tested; (d) validation consisting of a qualitative, visual inspection of the results of the tested systems; (e) selection of the system which performed the best; (f) runtime operation with the selected system, consisting of feature extraction, intermediary classification and final classification.

The approach is evaluated on two datasets acquired with different lidars and possessing different characteristics. The evaluation is done quantitatively using one of the datasets, and qualitatively using another. The approach was the subject of the following paper:

> Artur Maligo, Simon Lacroix. *Classification of Outdoor 3D Lidar Data Based on Unsupervised Gaussian Mixture Models*. IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2015. To appear.

1.4 Thesis Structure

Chapter 2 presents the state of the art. Chapter 3 explains the approach in detail. Chapter 4 describes the evaluation. Chapter 5 concludes the thesis with an overview and directions for future research.

Chapter 2

State of the Art

In this chapter, we present the state of the art of the core elements of this work. This is done with the goal of explaining the underlying concepts, putting in perspective the possible alternatives and presenting the main challenges involved. Section 2.1 presents the input of our system, lidar point clouds. Section 2.2 presents the output, semantic models. Section 2.3 presents the framework of probabilistic classification, the adopted means to compute semantic models from lidar point clouds. Section 2.4 concludes the state of the art and points to our approach.

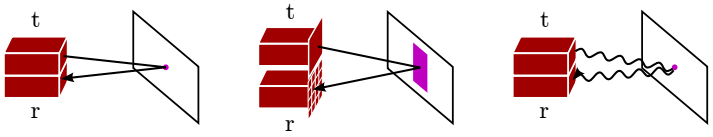
2.1 Lidar Point Clouds

In this section, we review, firstly, the technology behind lidars, then the issue of data structures for point clouds, and finally the lidar systems used for acquisition.

2.1.1 Lidar Technology

Lidar, also written LiDAR or LIDAR, stands for *light detection and ranging*. A lidar works by emitting a laser signal and using its reflection to calculate the *distance* between the sensor and the reached surface. The distance is expressed in the sensor's *reference frame*. The terms *depth* and *range* are also commonly used to refer to this measurement. There are a few different methods that may be used for distance computation. Lidar devices can be classified according to the employed method. A deeper explanation of the internal workings of lidars can be found in [McManamon, 2012].

Time-of-flight lidars (figure 2.1a) emit a laser pulse and use an electronic detection system capable of measuring the time between its emission and its detection [Borenstein et al., 1996]. The transmitter and receiver are located coaxially, or in close proximity, and therefore the pulse travels in an essentially straight line from the sensor to the object



(a) Time-of-flight (b) Flash (c) Phase-shift

Figure 2.1: *Lidar Methods.* The methods are illustrated in a schematic way. t indicates the transmitter, r the receiver. The different laser patterns are shown through the violet forms on the target surface. 2.1b shows the array of detectors of the receiver. 2.1c emphasizes how the signal is processed based on the laser wave phase.

and back. Because the speed of light is known, it is possible to compute the distance separating the sensor from the object as a function of the elapsed time.

Flash lidars (figure 2.1b) are a variant of time-of-flight lidars. They emit a laser pulse covering a large field-of-view, effectively illuminating the scene. The receiver is composed by a 2D array of detectors, the focal plane array, in a design analogous to a camera. Upon detection of the returned signal, each detector, through the time-of-flight principle, provides a measure of the depth of the corresponding area of the scene.

Another method worth mentioning is the *phase-shift measurement* method (figure 2.1c). It consists in emitting a continuous lidar signal, amplitude-modulated (AM), and then comparing it with the returned signal to compute the difference in phase. This difference is used to compute the distance travelled by the laser.

Besides distance, lidars might also return other measured quantities. A common one is the intensity of a detected laser signal. The intensity depends on the emission power of the lidar, on the reflectivity properties of the target surface, and on the incidence angle of the laser beam with respect to the surface.

A few problems may arise when dealing with lidars, due to the physics of these sensors and their interaction with the world. Some of these are presented in [Adams et al., 2011]. A first problem is the phenomenon of a *missed detection*, which happens when there is no valid return signal for an emission. Another problem is the production of an *outlier*, which happens when the sensor measurement has no meaningful link with an object in the world. A third problem is the *noise* present in the measurements. Noise may originate from different factors: the inaccuracy of calibration parameters, the distance between the sensor and the object, the imperfections in the computation of intensity and distance, and others.

2.1.2 Data Structures

Lidars must have a known geometric model, or calibration model, so that the distance measurements may be converted to 3D points expressed in Cartesian coordinates in the sensor's reference frame. Basically, such conversion can be done as a function of the measured distance, of the angles defining the line-of-sight of the laser beam, and of the sensor's calibration model. The final output of this process is a set of 3D points, grouped in a structure called *point cloud*.

There is, however, another structure which is commonly used to represent lidar data: the *range image*. In this case, the points are defined directly by the distance value, as well as by the angles of the line-of-sight of the laser beam: the *azimuth* angle and the *elevation* angle. The resulting data is organized as a 2D, image-like structure, where pixel values indicate depth. In many cases, the laser data is not dense enough to produce a complete image, where all pixels are filled, and methods that extrapolate distances to the missing pixels might be required.

Some works take into account the fact that a laser measurement also brings information about the free-space between the sensor and the object [King, 2008], and thus a measurement can be modelled as a segment, instead of just a point. Such representation leads to a *free-space model* of the scene.

A point cloud must be stored in a certain data structure. When the structure keeps points ordered in a way that reflects their positions in the scene, we say that the cloud is *organized*. Otherwise, the cloud is *unorganized*. An organized cloud is usually represented in the sensor reference frame, since this form allows it to be stored in a 2D matrix-like structure where the rows represent the elevation angles and columns represent the azimuth angles. The points can be expressed in their 3D coordinates, or through their distance value. The latter case, in fact, corresponds to the range image representation. Another form of creating an organized cloud is to build a neighbourhood graph, when the geometry of the lidar allows it, as done in [Moosmann et al., 2009].

Processing point clouds usually requires the computation of distances between points in order to find a pair or a group of close points, a problem known as *nearest neighbours search*. Searching for the neighbours of every point in a cloud may take a prohibitive amount of time, and an efficient search method must thus be applied. Organized point clouds offer a great advantage in this situation, because the neighbours of a point can be searched directly by examining points at adjacent rows and columns in the data structure.

Efficient search methods exist for unorganized point clouds too. In general, they consist in exploiting special data structures that ease the search task. Some of these data structures are tree-based, like the octree or kd-tree, as explained in [Rusu, 2009]. An octree is relatively fast to be built but relatively less efficient for the search, compared

to the kd-tree. The work of [King, 2008] proposes another tree structure: the spherical quad-tree. Other data structures are based on hashing, as done, for instance, in [Lalonde et al., 2006].

2.1.3 Lidar Systems

Generally speaking, there are two forms of acquiring data using lidars: *static* and *mobile*. Static acquisition involves placing a lidar sensor in a fixed position, usually with the help of a support. A mechanism moves the orientation of the laser beam after each measurement, allowing the sensor to gather points from the scene in all the programmed directions. The lidar remains fixed during all the scanning process. This scheme is appropriate for mapping small areas or specific objects, with high resolution and accuracy, and without time constraints.

Mobile acquisition, on the other hand, involves mounting a lidar device on a mobile platform, in such a way that data is collected all along the trajectory performed during the mission. Mobile platforms can be airplanes, boats, submarines, cars or other ground vehicles, robotic or not. Mobile scanning is suitable in cases where large areas must be mapped, or when time is an important constraint. An example is the case of a UGV that must navigate in the environment, and, for this, it must continuously model its surroundings to be able to plan its path.

Mobile acquisition used to be done with lidars that scan along a 2D plane. This has been changing since the Velodyne HDL sensors were introduced [Velodyne, 2007]. These sensors possess not only one, but an array of emitters, and also a rotating head. The combination of the firing of the emmiters and the continuous rotation of the head results in a scan that covers a 360° *horizontal field-of-view* around the sensor and a certain *vertical field-of-view*, instead of just a plane. For this reason, such lidars, in comparison to 2D lidars, allow the scanning process to be performed faster, and provide scans that are denser. The faster and denser scanning brought by these sensors has taken the mobile acquisition possibilities to new levels.

Whichever the acquisition system used, perceiving the environment with a lidar is subject to the following important challenges: *nonuniform sampling*, *occlusions* and *incomplete data*. All these arise from the combination of the sensor's' *sampling pattern* and the position of the objects relative to the sensor. Indeed, each lidar possesses a characteristic sampling pattern. Nonuniform sampling is produced by the projection of this pattern onto the objects in the world, located at different relative positions to the sensor. The decrease of the sampling density in function of the distance to the sensor, an important property of point cloud data, is a consequence of this phenomenon.

Occlusions are caused when an object cannot be fully perceived because it is located

behind another object along the field-of-view of the lidar. The same object is therefore perceived in different ways if the laser is moved to different positions. Lastly, the presence of incomplete data is related to the latter challenge. Incomplete data consists simply in the case where the sensor's point of view does not allow a full perception of the object, producing an ambiguous representation.

2.2 Semantic Models

Environment models provide the base on which rely other tasks of a robot. A model is composed by elements, the constituents of the model, and this elements define the *model structure*. The data used for modelling may come from any type of sensor, but here we consider models built with point clouds. One can distinguish two types of models: *geometric* and *semantic*. A geometric model contains elements with purely geometric information, such as occupancy or surface representation. A semantic model associates its elements with classes which are meaningful in some way for a considered *target task*.

A semantic model may be built on the basis of a geometric model simply by the assignment of classes to the geometric elements, or use a specific structure of its own. Classification comes in as the stage of semantic modelling where the elements are classified. Thus, the choice of the structure determines the type of element being classified, which in turn, impacts the classification process. Indeed, different elements encode the information about the environment in different ways. In this section, we review the applications of semantic models, then the structures of semantic models. We finish by introducing the problem of localization, inherent to any modelling task.

2.2.1 Applications of Semantic Models

An important application of semantic models is *terrain traversability analysis* [Papadakis, 2013]. A semantic model provides the capacity to distinguish zones in the environment according to traversability classes. Each class is associated with a cost, resulting in a cost grid that can be used by a robot to perform path planning, for example with the D* Lite algorithm [Koenig and Likhachev, 2002]. It should be noted that geometric models can also be used as a base for path planning tasks, but in this case with a continuous traversability metric, or some geometric criterium, instead of a discrete metric based on traversability classes.

Semantic models can also be found in applications involving *object detection and tracking*. The semantic information is represented in the detection step. Detection consists, essentially, in processing the input observation and classifying it as being the object-of-interest or not. In a case where many types of objects are considered, the input must be

classified into one of the types. The work of [Himmelsbach et al., 2009], for istance, tackles the case of vehicle detection in an urban scenario.

Detection of zones-of-interest can also be performed by means of semantic modelling. Applications such as search and rescue require the robot to interpret the environment and find affected zones, which are possibly partially destroyed and contain rubble, debris and maybe survivors. In fact, any application where the robot must interact with the environment might require such type of capacity from the robot.

Semantic models can help at solving *data association*. Examples of data association are place recognition [Granström et al., 2011], registration [Segal et al., 2009] and finding correspondences between features in mapping [Thrun, 2003]. Performing the search and matching processes at the semantic level leads to a reduction of the search complexity. This property is exploited in [Das et al., 2014], where registration is performed classwise, that is only between points belonging to the same class.

Another way of applying semantic models is as a visualization tool for humans. Such visualization capacity is useful, for example, in a case of situation awareness [Birk et al., 2009]. Indeed, models not only provide information to the autonomous robot itself, but also to the human clients involved in the robot's mission.

When semantic modelling consists in creating a map of the environment, we use the term *semantic mapping* [Nüchter and Hertzberg, 2008; Pronobis, 2011]. Not all semantic models are semantic maps. As an example of this distinction, we can take a case of object detection and tracking, where the individual point clouds are processed in sequence, but the output models are not integrated into a single, global model of the environement, a map. In this case, the individual models are semantic because they represent the detected objects, but there is no map construction.

Whichever the application, semantic modelling faces the problem of the *environment variability*. This is especially true for the case considered in this thesis, the case of outdoor environments. In outdoor environments, objects of a same type may present a great variety of shapes and scales. Differences in shape and scale are multiplied when we consider objects of different types. In addition, the environment might contain an important number of objects types. One of the main challenges of semantic modelling is to try to capture all this variability.

2.2.2 Structures of Semantic Models

Point clouds can be considered as the simplest model structure. In pointwise classification, classification is applied directly to 3D points [Behley et al., 2012; Brodu and Lague, 2012; Lalonde et al., 2006]. Only local information, that is information about the neighbourhood of a point, is used for classification. Therefore, no assumptions regarding the segmentation

of the points are made, making this approach agnostic with respect to shapes.

A grid representation of the environment is a common type of structure. An example of a grid-based geometric model is the *occupancy grid*. Basically, such models consist in grids whose cells can be occupied, free, or in an unknown state. The basic, 2D version is a projection of the world onto a 2D grid of occupancy cells. When the environment is too complex to be represented in 2D, a 3D occupancy grid can be applied. In this case, we speak of a *volumetric model*. Such models can be extended to multi-resolution versions by using tree structures such as quadtrees in the 2D case and octrees [Hornung et al., 2013] in the 3D case.

Semantic grid-based models extend the previous models through the classification of the grid cells. Here, classification takes into account the data points lying inside the cells. In comparison with pointwise models, these models impose a regular grid structure to the environment. A consequence is that regions of the world which do not fit into a grid structure may be poorly represented. An example could be a scene where the corner of a building lies next to a road, and both entities are located inside a single cell. The cell in case is ambiguous, because it represents different entities.

Other model structures represent surfaces through *polygon meshes*. A widely used form of mesh is the triangular mesh. Alternatively, surfaces can be approximated by a set of convex planar polygons, known as *convex hulls* [Rusu, 2009]. The mesh itself constitutes a geometric model, and its construction corresponds to the problem of *surface reconstruction*.

The semantic versions of polygonal models aim at finding the objects underlying the mesh. In [Triebel et al., 2012], the point clouds are first converted to a triangular mesh. The mesh is then segmented and the segments, classified. The use of a mesh has an important consequence: it allows the sampling of points from the mesh, which makes it possible to generate uniform samples for subsequent feature computations.

Another important class comes in the form of models based on 3D *geometric primitives* such as plans, cylinders, spheres, cones and so on. These primitives serve as approximations of the objects' shapes and surfaces. The problem of matching primitives to a set of points is known as *model fitting*. It is interesting to note that here the distinction between a geometric or a semantic model becomes less clear. Primitives can be used only to provide geometric information, but we can rightly think of them as classes too, given that a robot might have a specific form of reaction associated to each primitive type, which shows that a primitive carries a meaning with it.

Some approaches apply segmentation on the points and then use the segments as classification targets [Himmelsbach et al., 2009; Moosmann et al., 2009; Moosmann and Sauerland, 2011]. Segmentation constitutes an alternative to geometric models, in the sense that the resulting segmentation structure could not be considered as geometric, but

provides nevertheless the base for a subsequent semantic model. Segmentation permits the use of global information in the classification, that is information about the whole object. This approach allows for a richer description of objects, but it introduces the constraint of dealing with all the variety of shapes.

There are methods that consider a specific form of segments: voxels [Aijazi et al., 2013; Lim and Suter, 2009]. In these works, points are grouped into voxels of adaptive sizes, then a subsequent segmentation step is applied, resulting in super-voxels, which are the targets of classification.

The ideal structure for a semantic model lies in a trade-off between a structure that encodes a sufficient amount of information and a structure that is able to properly segment the objects in the environment. In other words, an element of the model should correspond to a single entity, a single class. The more simple the structure, the more generic it is. Pointwise models are at this end of the trade-off. More complex models may provide better representations for the objects in a scene, but at the risk of including different objects under the same element. In all the cases discussed, an important challenge that must be faced is *clutter*.

2.2.3 Localization

As previously mentioned, one of the forms of acquiring data with lidars is mobile acquisition. When it is applied to a *mobile mapping* task, an important problem that arises is *localization*. Let's examine the case where a UGV is equipped with a lidar, scanning the environment while it moves. The points returned by the sensor are expressed in the sensor reference frame. The sensor itself is positioned in a certain way on the robot, so that its reference frame is linked to the robot reference frame by means of a *transformation*. Finally, the robot is in a certain position with respect to an original, global reference frame. This position corresponds effectively to the transformation between the robot and the global reference frame. By applying the proper transformation, points in a reference frame can be represented in another frame. This is illustrated in a schematic way in figure 2.2.

Lets return now to the map that must be built by the robot in the mission. Mapping implies that the sensor point clouds, initially given in the sensor frame, must be converted to the global frame, which is taken to be the map's frame. In order to do so, the transformation between the sensor and the robot frame must be known, as well as the transformation between the robot and the global frame. The first transformation is usually known, since it is given by the robot's setup. Estimating the second one constitutes the problem of localization.

Mobile acquisition, therefore, when used for mobile mapping, comes with the require-

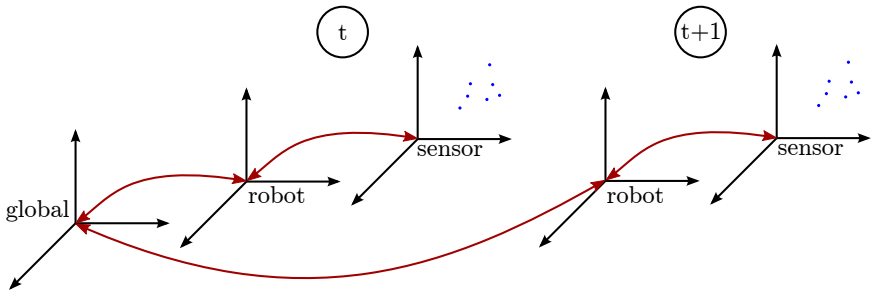


Figure 2.2: *Mobile Acquisition and Reference Frames.* The robot reference frame appears at different positions, at instants t and $t+1$, with respect to the global reference frame, indicating that the robot has moved. The sensor reference frame changes accordingly. The points acquired with the sensor are initially expressed in the sensor frame. Through frame transformations, depicted by the arrows, we can express the points in other reference frames.

ment of having an accurate, high-rate localization. The time constraints imposed by the mobile mapping task are translated into time constraints for the localization task. Ideally, the robot must know its position at the acquisition time of every observation. For a mobile lidar such as a Velodyne sensor, for example, the localization must run at a very high frequency.

In the case of errors in the estimation of the robot's position, the acquired points will be erroneously integrated into the map, making it inaccurate. Inaccuracies generated in this way may be called *localization noise*. The noise affects the quality of both the geometric and the semantic information that might be contained in the map.

2.3 Probabilistic Classification

Classification can be accomplished through the framework of machine learning [Bishop, 2006], or simply learning. In the standard classification problem, an input piece of data must be assigned to a class. Often, the input data cannot be directly used for classification, and it is first necessary to extract from it information in an exploitable format. We refer to the extracted information as *features*, and we refer to the process as *feature extraction*. An arbitrary number of features can be used. The space containing all the possible input data is the *input space*, and the space containing all the possible feature values is the *feature space*. Through feature extraction, the input piece of data, which is a point in the *input space*, is thus mapped to a point in the *feature space*.

Classification is divided into two steps: *inference* and *decision*. Let $\boldsymbol{x}$ be an input point in the feature space, that is, an input point after feature extraction, and c_i be the i-th

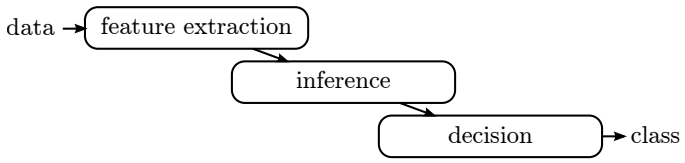


Figure 2.3: *Classification.* Firstly, an input piece of data goes through feature extraction. Then, the resulting feature point is fed to the inference process. Finally, the inference result is used as a base for decision, which outputs the class that should be assigned to the input data.

class of the *class set* $C = \{c_1, \ ..., \ c_{N_C}\}$, with N_C being the number of classes in C. Let y_i be a variable indicating the assignment of class c_i. This allows the distinction between the class and the variable representing it. Inference consists in calculating the *class posterior probability* $p(y_i|\boldsymbol{x})$ for each class. It is performed by a *classification model*, or *classifier*. Decision, in turn, consists in analysing the posterior probabilities and assigning a class to $\boldsymbol{x}$. The processes of feature extraction, inference and decision can be viewed in figure 2.3.

Learning comes into play when we need to instantiate the stages of feature extraction, inference and decision. A full learning process is usually divided into *training*, *validation* and *test*. Learning is based on an input dataset, and on input prior information concerning the stages of the classification pipeline: the features, the classifier, the decision method. In fact, the dataset itself can be seen as part of the prior. In the best case, all the elements of the pipeline are completely defined, with the exception of the classifier's *parameters*. Finding these parameters is the goal of training. The information guiding the training phase comes from the dataset, which in this case is called the *training set*.

It turns out, however, that there are more parameters to be determined. Starting in feature extraction, we may need to select a certain number of features that will be used among a larger set of available features. Moreover, each individual feature may depend on a list of parameters that must be set. Making these choices corresponds to the problem of *feature selection*. Concerning the classifier, the situation is similar. A set of models can be considered, each one depending on its parameters, but also on its *complexity parameters*, or *hyperparameters*. The task of finding the most appropriate model among all the choices is known as *model selection*. Finally, the decision model might also depend on a set of parameters to be completely defined.

To compute the remaining parameters, many methods can be applied. Feature and model selection, for instance, are full research topics on their own. However, a basic approach is to use the validation stage to this end. In this way, the classifier's parameters are computed during training, as previously explained, while all of the other parameters are computed in validation. Here, part of the dataset must be reserved to be used in this

step, constituting the *validation set*.

Once training and validation are over, the full classification system is defined. Its performance is then assessed in the test phase. Testing requires its own share of the dataset, the *test set*. The complete learning dataset is thus split into training set, validation set and test set. The reason for using different data in each case is that the system must be able to classify new data, data which is different from that used to find the parameters. The ability to do so is called *generalization*, and is a core characteristic of learning-based systems.

Classifiers can be distinguished between *discriminant functions*, *discriminative models* and *generative models*. A discriminant function is a function that directly assigns a class to the input, $\boldsymbol{x}$. Such functions correspond to a case where inference and decision are merged together and there are no intermediary probabilities. Without probabilities, such classifiers loose in flexibility and in complexity, and therefore are only able to tackle relatively simple problems.

Discriminative and generative models, on the contrary, are probabilistic. Their output, as explained before, consists in the class posterior probabilities, $p(y_i|\boldsymbol{x})$. Discriminative models represent these probabilities directly. Generative models represent them through individual terms, according to the Bayes' equation:

$$p(y_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y_i)p(y_i)}{p(\boldsymbol{x})}. \tag{2.1}$$

$p(y_i)$ is called *class prior probability*, and $p(\boldsymbol{x}|y_i)$ *likelihood*. $p(\boldsymbol{x})$ is the *normalization term*, and can be computed as

$$p(\boldsymbol{x}) = \sum_j p(\boldsymbol{x}|y_j)p(y_j). \tag{2.2}$$

The term generative reflects the fact that knowing the distribution $p(\boldsymbol{x})$ allows us to generate samples of $\boldsymbol{x}$.

After inference, performed by such probabilistic classifiers, we proceed to decision. In its most basic form, decision is simply done by selecting the class with the highest posterior probability.

The factorization of the posterior into likelihood and prior constitutes an important characteristic of generative models. Such factorization can be viewed as an extra knowledge which is put into the model, that is, knowledge about its internal structure. However, it can also be viewed as an additional assumption which must be made regarding the model. If the assumption does not correspond to the real situation, the classifier's accuracy is impacted. Thus, working with discriminative or with generative models is a matter of how much information, or knowledge, we possess about the model.

The classifiers discussed so far are called *parametric classifiers*. There is, however, another type: *nonparametric classifiers*. Two common examples are the *kernel classifiers* and the *nearest-neighbour classifiers*. Nonparametric methods, as the name indicates, take a different approach, where the points in the dataset constitute the classification model itself, and thus there are no parameters to be trained.

The advantage of nonparametric classifiers is that, because they use directly the structure present in the dataset, they are able to model complex distributions, as complex as the dataset, in fact. The disadvantage is that the dataset must be always available, and that accessing and searching the data in it might be slow. Nonparametric models are used when we do not have enough information about the problem, or when the problem is complex. In these cases, such models compensate for the lack of information by using the richness of information present in the dataset.

2.3.1 Supervised Learning

Another important distinction that must be made regards *supervised learning* and *unsupervised learning*. In supervised learning methods, the classes have already been defined and assigned to the points in the dataset. Therefore, learning, in this case, assumes the goal of finding the classification system that best generalizes the class assignments present in the training set.

Supervised learning is frequently applied in 3D data classification. A comparison is presented in [Behley et al., 2012]. [Brodu and Lague, 2012] uses linear classifiers, [Himmelsbach et al., 2009] uses a SVM, [Lalonde et al., 2006] uses a GMM and [Lim and Suter, 2009; Munoz et al., 2009] use a CRF.

The crucial characteristic of such methods is that they work with predefined classes whose *semantic interpretation* is already known. The disadvantage is that they require manual labelling of the data by a human domain expert, a process which does not scale well with respect to the amount and to the complexity of the data.

2.3.2 Unsupervised Learning

Unsupervised learning methods do not work with predefined classes. Instead, they are free to find the patterns that can be encountered in the training set, and come up with the most appropriate classes to represent these patterns. The predefined classes in the supervised learning context represent an additional prior information which is assumed about the problem. By not making such assumption, unsupervised methods adapt to the data in a natural way. We may say that unsupervised learning is data-oriented.

The use of unsupervised learning is relatively less common. The work of [Moosmann and Sauerland, 2011] presents a method where 3D points are segmented and the resulting

segments are used for the unsupervised discovery of classes. In [Ruhnke et al., 2010], an unsupervised method based on range image features is used to generate a set of words, which are in turn used to replace similar regions of a map to compress its size. The work of Steder et al. [2011] applies k-means clustering to range image features in order to assist in the place recognition problem.

The most important problem in this approach is that the output classes must then be semantically interpreted or, in other words, be given meaning in the context of the target task, which may not be always possible. Thus, the interpretation still requires supervision. The key advantage, in this case, is that no manual labelling of the dataset is required.

2.3.3 Feature Extraction

We begin this section by making a remark concerning the use of the term feature. In the literature, it is possible to find many related terms but that are not necessarily equivalent, like descriptors, signatures and attributes. Here, we consider a feature as the input or as one of the inputs given to the classifier. In this work, since point clouds are used as the data for the classification task, features assume the format of the geometric and statistical characteristics of the scene represented by the cloud.

The choice of features is crucial in a classification problem. Ideally, a feature should be able to generate a unique description of each class. The more descriptive the features are, the less complex the classifier needs to be [Behley et al., 2012]. In the rest of this section, we discuss some important choices that must be made when computing features. We explain the consequences of these choices with respect to some known challenges, previously mentioned, such as changes in the relative position of object and sensor, nonuniform sampling, occlusions and clutter.

Support region type. Features are normally computed using the points contained in the neighbourhood, or support region, of a query point. In pointwise classification, the neighbourhood, can be determined either by the number of nearest neighbours of the query point or by the size of a region around it [Behley et al., 2012]. In the second case, the region can be a sphere, a voxel or other type of 3D element. The choice of the support region depends on the application and on the geometric characteristics of the sensor being used. Some sensors, like the Velodyne, provide a point cloud which is relatively sparse. In this case, using a support region with a fixed number of neighbours means that features are being computed in areas of varying sizes, which greatly alters the results. On the other hand, using a support region with a fixed size means that features are being computed in areas with a varying number of points, which also influences the results.

Keypoints. Sometimes feature computation is preceded by a search for keypoints. Keypoints are the points in a cloud whose neighbourhood exhibit a special property. For this reason, they are used in the feature computation step, whereas the other points are ignored. The neighbourhood of a keypoint is called a region of interest. One of the properties that can be searched for is stability in orientation [Quadros et al., 2012]. A planar region is well characterized by its normal vector orientation, and therefore features that use this orientation to determine its reference frame are viewpoint invariant. Additionally, since features are being computed only at a subset of the points in a cloud, the time spent in the process is reduced.

Feature parameters. The design of a feature involves setting some parameters. An important example is given by the frequently used histogram features, as shown in [Behley et al., 2012]. Histogram parameters, like the bins' size, must be chosen in order to reflect the environment properties that are being evaluated. Histograms with larger bins lead to features that can only describe coarser characteristics of the environment, while histograms with shorter bins lead to features that capture the details, but that are therefore more affected by noise in the data. The sampling density is an essential factor to be considered when choosing the bins' size. It is useless to try to describe and to compare poorly sampled surfaces using histograms with many short bins and, in the same manner, it is useless to use histograms with few and large bins to represent richly sampled surfaces.

Reference frame. In order to be matched and compared, features must be invariant to the sensor's viewpoint. In other words, it should ideally be possible to represent an object of the environment in a way that does not depend on its position with respect to the sensor. Therefore, features must be designed to be viewpoint invariant, or else techniques to align them must be applied [Quadros et al., 2012]. It is common either to align features to the z axis of the global reference frame, in order to ensure that they are expressed in a stable reference frame, or to align features to the z axis of the region's local reference frame, in order to express them using the local geometry. This choice depends on the nature of the classes that are being considered. Objects like cars and trees, which have a strong vertical characteristic with respect to the ground, are well described in the global reference frame. Other objects, like vegetation or surfaces with arbitrary orientations, can be well represented in their own reference frame.

Scale. Besides choosing the support region type, it is necessary to define its parameters, which can be either the number of neighbours or the region's size. This choice depends on the scale of the properties that the features are trying to encode [Behley et al., 2012; Lalonde et al., 2006; Quadros et al., 2012]. When considering small support regions, a

car's surface looks exactly like a flat ground's surface. However, when considering big support regions, a car's appearance might be affected by nearby vegetation. Thus, the size is important when dealing with cluttered environments. Moreover, by using bigger support regions the occurrence of occlusions is increased. With respect to the number of neighbours, using few points leads to features that are sensitive to noisy measurements, while using many points leads to features that are less precise, because details are filtered in the presence of a large number of points.

Considering pointwise classification, a standard method is, given a target point, to take all points lying inside a spherical support region centred around it, and use these in the feature computation [Behley et al., 2012; Lalonde et al., 2006; Steder et al., 2011]. Given that a sphere radius is specified, the resulting feature only provides information about the point neighbourhood on the specified scale. This method is not efficient when the classes present in the environment are characterized by different scales.

To overcome the problem mentioned above, multi-scale methods have been proposed. In [Unnikrishnan, 2008], an adaptive process is performed: the radius of the support region is chosen based on the shape of the neighbourhood. This method is however computationally expensive.

Another multi-scale approach was proposed in [Brodu and Lague, 2012]. In this work, multiple spherical support regions, with different radii, are used simultaneously for feature extraction. The resulting vector is a combination of the feature values extracted at the different radii, and thus encodes how the shape of the point's neighbourhood is perceived at different scales.

Neuhaus et al. [2009] presents a hierarchical approach for dealing with multiple scales. A point cloud is firstly analysed as a whole. If it is not considered flat according to their criterion, it is divided in halfs, following a 2D grid model. These halfs, which are 2D cells, are then submitted to the same analysis. This procedure continues in a recursive manner, and the division terminates if a cell is considered flat or if it has reached a minimum size.

Works applying segment classification deal with the scale problem in an implicit way, because segments assume different sizes depending on the object being segmented [Aijazi et al., 2013; Himmelsbach et al., 2009; Moosmann and Sauerland, 2011; Lim and Suter, 2009].

2.4 Conclusion

Pointwise classification has the advantage of not biasing the classification by introducing a segmentation. It is agnostic with respect to shapes, using only local information about a point's neighbourhood. Our approach adopts this method. We aim at avoiding the manual labelling of datasets and at using a classification model capable of naturally handling

various types of data. We choose for this an unsupervised GMM. To provide the classes with a semantic interpretation, we add a supervised grouping as a second layer to the classification model. Regarding the feature extraction, besides considering a single spherical support region, we also explore the method of using multiple regions simultaneously, found in [Brodu and Lague, 2012] and previously discussed.

Chapter 3

Classification

Our approach relies on the proposed two-layer classification model. We perform pointwise classification, such that a point, associated with its support region, or neighbourhood, is the element being classified. In the multi-scale case, a point is characterized by multiple neighbourhoods. The classification model is composed of two layers. The intermediary layer consists of a GMM. This layer provides the intermediary classes. The final layer consists of a grouping of the intermediary classes into the final classes, which are the output of the system.

The whole system consists of the stages of feature extraction, intermediary classification and final classification. Looking at the standard classification stages of feature extraction, inference and decision, our feature extraction stage corresponds to the standard feature extraction stage, while our intermediary classification corresponds to the inference and decision stages. Indeed, the intermediary classification performs both operations in sequence. Our final classification can be seen as an extra grouping stage. Figure 3.1 illustrates the classification system.

Sections 3.1, 3.2 and 3.3 explain the stages of feature extraction, intermediary classification and final classification, respectively. Section 3.4 explains the learning process used to obtain a full, definitive form of the classification system.

3.1 Feature Extraction

The feature extraction process is performed pointwise. In the single-scale case, it takes into account a target point and the points in its spherical neighbourhood of radius r. In the multi-scale case, it takes into account multiple spherical neighbourhoods, determined by a set of radii $R = \{r_1,\ ...,\ r_{N_R}\}$, N_R being the number of radii. Three values are computed for each scale, which leads to a feature vector $\boldsymbol{x} = [x_1\ x_2\ x_3]^T$ with dimension 3, if single-scale, and to a feature vector $\boldsymbol{x} = [\boldsymbol{x}_1^T\ ...\ \boldsymbol{x}_{N_R}^T]^T$ with dimension $3N_R$, if multi-scale. In

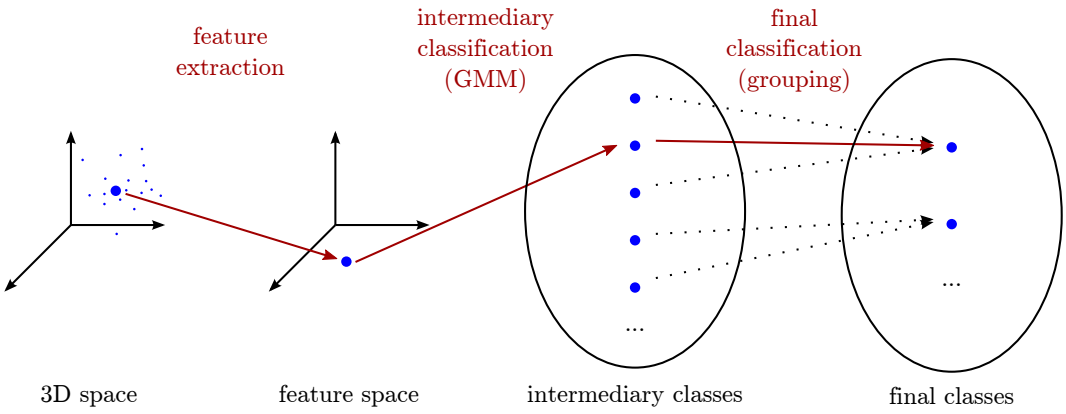


Figure 3.1: *Classification System.* A target point in the 3D space is transformed to a point in the feature space through feature extraction. Then, this point is classified into an intermediary class by the GMM, and finally this class is mapped to a final class according to the grouping.

the latter case, $\boldsymbol{x}_i$ indicates the feature values computed at radius r_i.

The input point cloud is assumed to be expressed in the sensor reference frame. Moreover, for the computation of the third feature value, the transformation to the world reference frame is necessary. This transformation is assumed to be given. Thus, the inputs of feature extraction are actually a point cloud and its corresponding sensor-to-world transformation. The reason behind these requirements will be made clear in the remaining of the section.

The three feature values result from a Principal Component Analysis (PCA) operation applied to the target point's neighbourhood. PCA works under the assumption that the distribution of points in the neighbourhood is Gaussian. Its goal is to find the local orthonormal basis whose axes are the directions of maximum variance, the principal components, of the distribution.

Let $Q = [\boldsymbol{p}_1 \ ... \ \boldsymbol{p}_{N_Q}]^T$ be the matrix composed by the N_Q points $\boldsymbol{p}_i$ in the target point's neighbourhood. One way to perform PCA is to start by computing the sample mean and covariance of the points:

$$\boldsymbol{\mu}_Q = \frac{1}{N_Q}\sum_{i=1}^{N_Q}\boldsymbol{p}_i, \quad \Sigma_Q = \frac{1}{N_Q}\sum_{i=1}^{N_Q}(\boldsymbol{p}_i - \boldsymbol{\mu}_Q)(\boldsymbol{p}_i - \boldsymbol{\mu}_Q)^T. \tag{3.1}$$

The covariance matrix is then decomposed in order to find its eigenvalues and eigenvectors. Because it is symmetrical and positive-semidefinite, specialized and efficient decomposition methods can be used. The eigenvalues and eigenvectors constitute the result of PCA: the order of the eigenvalues indicate the order of the principal components and the eigenvectors indicate the components themselves.

The knowledge about the points' distribution brings with it information about the local surface shape. Numerous works on 3D lidar data processing exploit this property. [Brodu and Lague, 2012] uses the normalized eigenvalues at multiple scales to describe the dimensionality of the shape. [Lalonde et al., 2006; Munoz et al., 2009] use the differences between the eigenvalues to this end. [Moosmann and Sauerland, 2011; Triebel et al., 2012] use ratios, instead of differences. [Neuhaus et al., 2009] uses the eigenvalue of the most vertical eigenvector to evaluate flatness. Much of this work is inspired by [Guy and Medioni, 1997], that used the principal components under the framework of voting. Other works applying PCA under this framework are [King, 2008; Stumm et al., 2012]. Our approach, in turn, builds on the multi-scale PCA features found in [Brodu and Lague, 2012], as we explain hereafter.

Let $\lambda_1 > \lambda_2 > \lambda_3$ be the eigenvalues output by PCA, and $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ the eigenvectors. As done in [Brodu and Lague, 2012], we can take the following values as the first two feature values:

$$\tilde{x}_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}, \quad \tilde{x}_2 = \frac{\lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}. \tag{3.2}$$

The normalization of the eigenvalues makes it possible to discard one value, and thus one dimension, without loosing in description power. These two values encode the shape of a distribution of points, or more especifically, its dimensionality, as shown in figure 3.2. Considered separately, the first feature distinguishes between 1D and 3D shapes, while the second feature distinguishes between 2D and 1D shapes. Taken together, they distinguish between 1D, 2D and 3D shapes.

Another form to exploit PCA is to interpret it as a plane fitting operation, as explained in [Klasing et al., 2009]. Through this point of view, the eigenvector $\boldsymbol{v}_3$, associated to the smallest eigenvalue λ_3, represents an estimation of the surface normal. The orientation of the normal is, however, ambiguous. Here, we use the fact that the cloud is in the sensor reference frame, and flip the normal in function of the viewpoint, which is the frame's origin. It is then possible to use the sensor-to-world transformation to transform the vector into the global reference frame, resulting in the global normal $\boldsymbol{n} = [n_x \; n_y \; n_z]^T$. The third feature value is given by the z coordinate:

$$\tilde{x}_3 = n_z. \tag{3.3}$$

In the 3D space, it makes no sense applying PCA on a set with less than four points, because such points will always be collinear or coplanar. Four points, on the contrary, can either be collinear, coplanar, or none of both, and thus can characterize arbitrary 3D shapes. Thus, during feature extraction we leave out points for which the condition $N_Q < 4$ holds. Such points are then also excluded from the classification. This situation occurs with higher frequency in the furthest regions of scans, where the laser sampling is

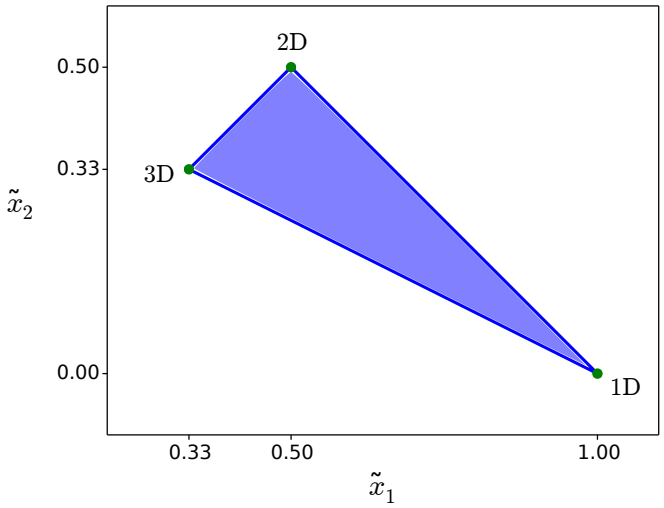


Figure 3.2: *Eigenvalue Features.* This is the space generated by the first two feature values, which correspond to the normalized first two eigenvalues output by PCA. The feature values lie inside the closed triangle. The edges of the triangle, $[1.0\ 0.0]^T$, $[0.5\ 0.5]^T$ and $[0.\overline{3}\ 0.\overline{3}]^T$ correspond to the cases where the 3D points assume pure 1D, 2D or 3D shapes, respectively. The definitive feature space is obtained after standardizing, and correspond to a translated and scaled version of this triangle.

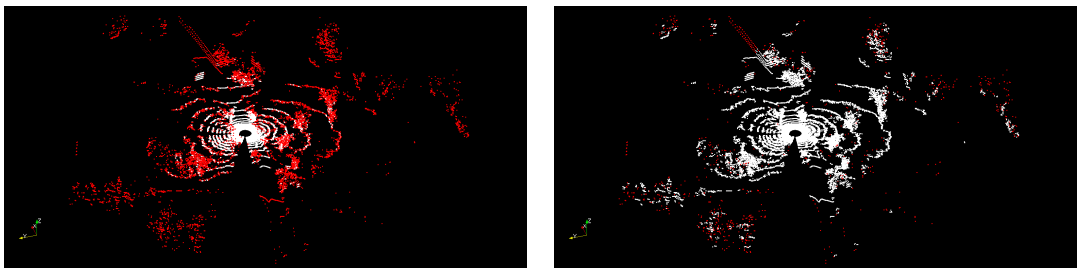


Figure 3.3: *Points Discarded in Feature Extraction.* At the left, the radius used is $r = 0.2m$, while at the right, $r = 1.0m$. The discarded points are shown in red. There is a large number of discarded points at the left, due to the sampling sparsity of the scan and the small scale. Using a large scale, as in the scan at the right, is able to greatly improve this situation.

more sparse. A beneficial consequence is that isolated outliers are naturally filtered out from classification.

The filtering of points also poses a problem: at small scales, due to the sampling sparsity, an important amount of points may be discarded. This situation is shown in figure 3.3. Moreover, other factors, such as occlusions and missing data, also contribute to this situation. A solution applicable when using the multi-scale features is to fill the values at the missing scales with the values coming from the next available larger scale. This operation is proposed in [Brodu and Lague, 2012]. It implies the assumption that a surface will not change when perceived at a smaller scale, which is an approximation. Indeed, it is a kind of smoothing operation, since at a larger scale, less details are perceived. In the cases where there isn't any larger scale available, the point is filtered out.

The feature extraction process concludes with a standardizing step. Features are compared on the basis of a distance metric. Standardizing aims at ensuring that every feature dimension contributes equally to the metric. In our case, we apply a statistical standardizing, relying on a mean μ_i and on a standard deviation σ_i for each feature dimension i. As we will discuss in section 3.4, these values are determined during training. For every point $\boldsymbol{x}$, for every dimension i, standardizing is applied in the following manner:

$$x_i = \frac{\tilde{x}_i - \mu_i}{\sigma_i}. \tag{3.4}$$

Parts of feature extraction were implemented using tools such as *Eigen* [Guennebaud et al., 2013] and *Point Cloud Library* (*PCL*) [Rusu and Cousins, 2011].

3.2 Intermediary Classification: GMM

The intermediary classification layer is a GMM. The GMM is a member of the family of mixture models, which as the name indicates, are models composed by mixtures of distributions. The basic goal of such a model is to represent a probability distribution, likely a complex one, by means of mixing multiple distributions. The individual distributions are called the model components.

The use of components has an important implication: it allows us to introduce the notion of membership, that is, we assume that each observed point belongs to a single component. This relation is captured by the introduction of a latent variable denoting the component to which a point belongs. We can then view the component as the class of the point, and use the model to perform classification. We exploit GMMs under this assumption.

Through feature extraction, a 3D point belonging to a point cloud is associated with a point $\boldsymbol{x}$ in the feature space. A GMM represents the distribution of $\boldsymbol{x}$ over the feature space by employing Gaussian distributions as components [Bishop, 2006]. Let $CY = \{cy_1,\ ...,\ cy_{N_{CY}}\}$ be the set of intermediary classes, N_{CY} being the number of classes. The component, or class, is indicated by the latent variable $\boldsymbol{y} = [y_1\ ...\ y_{N_{CY}}]^T$. This is done in the following manner:

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{y}} p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{y}} p(\boldsymbol{y}) p(\boldsymbol{x}|\boldsymbol{y}) = \sum_{i=1}^{N_{CY}} p(y_i) p(\boldsymbol{x}|y_i) = \sum_{i=1}^{N_{CY}} \pi_i \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i). \tag{3.5}$$

We note that $\boldsymbol{y}$ is a vector, and that we use y_i to denote the case where $y_i = 1$ and $y_j = 0$ for $j \neq i$, meaning that class cy_i is assigned to $\boldsymbol{x}$. Each class is Gaussian, and is defined by the following parameters: the mixing coefficient π_i, the mean $\boldsymbol{\mu}_i$ and the covariance Σ_i. We can also note, by the equation, that a GMM makes the following assumptions:

$$p(y_i) = \pi_i, \quad p(\boldsymbol{x}|y_i) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i). \tag{3.6}$$

Having in hands the distributions $p(y_i)$, $p(\boldsymbol{x}|y_i)$ and $p(\boldsymbol{x})$, we can compute $p(y_i|\boldsymbol{x})$:

$$p(y_i|\boldsymbol{x}) = \frac{p(y_i)p(\boldsymbol{x}|y_i)}{p(\boldsymbol{x})} = \frac{\pi_i \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i)}{\sum_{j=1}^{N_{CY}} \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \Sigma_j)}. \tag{3.7}$$

This is the Bayes equation, as mentioned in chapter 2. Moreover, as explained in the same chapter, we call $p(y_i)$ the class prior probabilities, $p(\boldsymbol{x}|y_i)$ the likelihood and $p(y_i|\boldsymbol{x})$ the class posterior probabilities. In the context of mixture models, the posterior is also called the component responsibility. Since a GMM is able to model the likelihood, it is a generative model, and because it models a distribution through components, it is parametric.

The computation of the posterior distribution corresponds to the inference step of classification. This is, therefore, how a GMM is able to perform classification. In our case, having obtained the posterior distribution through inference, we perform the decision step in sequence, and assign to a point the class that obtained the highest posterior probability. Indeed, inference and decision are both done in this intermediary layer of the classification model.

The GMM intermediary classes are data-oriented. They serve to capture all the different patterns that may be encountered. Ideally, if the model were powerful enough, it should be able to capture, to abstract the different environmental and sensorial factors influencing the perception. By environmental factors, we refer to the variability and the clutter present in the environment, while by sensorial factors, we refer to the perception effects derived from the sensor sampling pattern, as presented in chapter 2.

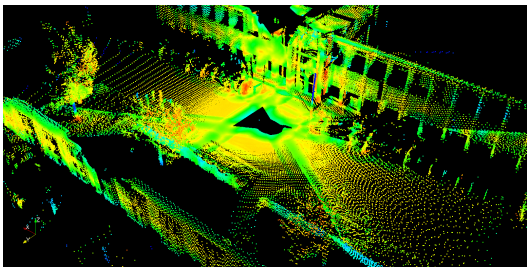
3.3 Final Classification: Grouping

The final classification layer is a grouping of the intermediary classes into final classes. Viewing it purely through the point of view of classes, the set of intermediary classes is denoted by $CY = \{cy_1,\ ...,\ cy_{N_{CY}}\}$, while the set of final classes is denoted by $CZ = \{cz_1,\ ...,\ cz_{N_{CZ}}\}$. N_{CY} and N_{CZ} respect the condition that $N_{CZ} \leq N_{CY}$. This operation aims at giving a single semantic interpretation to multiple intermediary classes. The semantics are ideally connected to useful properties in a target task. We say thus that the final classes are task-oriented.

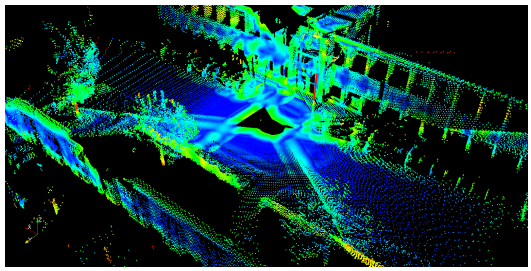
As an example, consider a case where the robot must distinguish the ground in its surroundings. The GMM might employ many classes to capture the distribution of ground points, as well as of non-ground points, but these intermediary classes are grouped into a set of two final classes, $CZ = \{ground,\ non\text{-}ground\}$.

The main limitation of this method is that, in fact, not all the intermediary classes can be exploited. Some of them correspond to objects of different nature, and thus cannot be grouped into a meaningful final class. In this case, the class is marked as *unknown* final class. The *unknown* points do not contribute to the resulting semantic model. In a way, this situation is analogous to the case where, in the decision stage, we refrain from classifying a point, which is done based in some uncertainty criterion. In our system, as explained, the decision is performed in the intermediary layer, but no uncertainty criterion is applied: all the points are classified. The unknown classification is brought over in the final layer, through the *unknown* class.

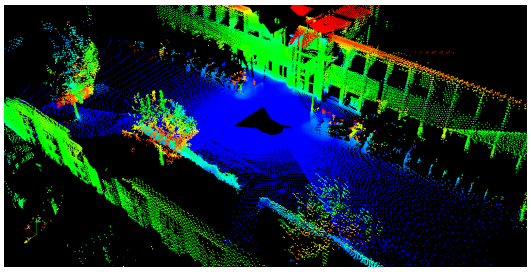
A property of this classification method is that a final class may be composed by an arbitrary number of intermediary classes. This number is an indicator of how many different patterns of the final class are encountered in the data. This is in constrast to the



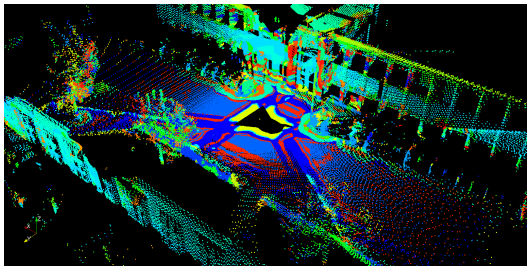
(a) x_1



(b) x_2



(c) x_3



(d) Output labels

Figure 3.4: *Intermediary Classification.* Figures 3.4a, 3.4b and 3.4c show the feature extraction results. Figure 3.4d shows the intermediary classification result. Here, $N_{CY} = 50$. In all scans, colours range from red, the lowest value, to blue, the highest value, indicating either the feature values or the class labels. Note how the intermediary classes apparently follow the patterns encountered in the features.

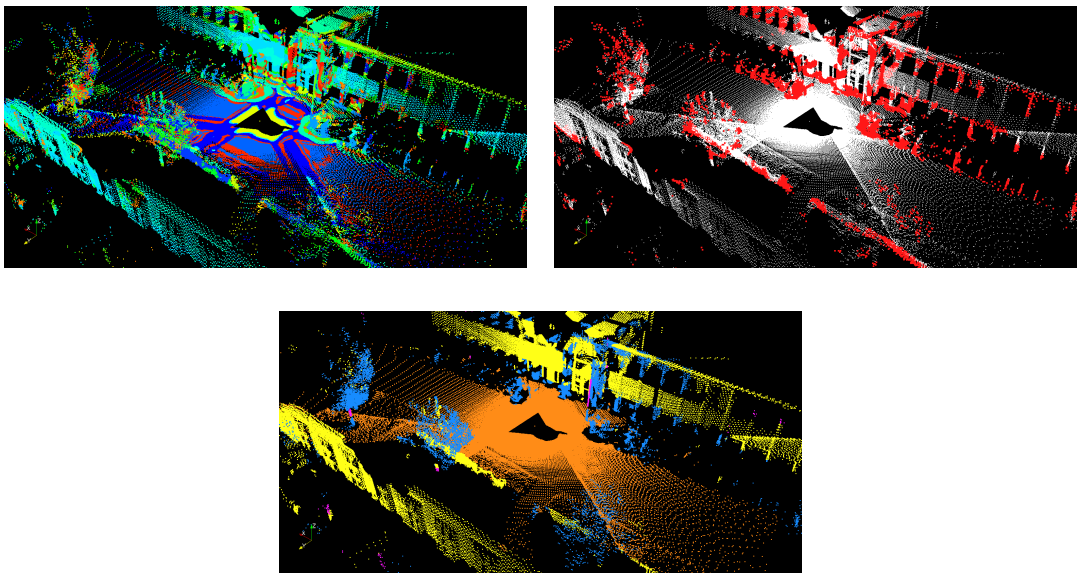


Figure 3.5: *Final Classification.* At the top-left, the result of the intermediary classification. At the top-right, the points classified as *unknown*, marked in red. At the bottom, the output of the final classification, coloured as: (orange, *ground*), (yellow, *wall-building*), (pink, *pole-trunk*), (blue, *vegetation-bicycle*).

standard supervised GMM, where the number of components is the same for every class. An example of final classification is shown in figure 3.5.

3.4 Learning

The learning of the full classification system follows the process of training, validation and test, as explained in chapter 2. A schematic overview of this process, as applied in our case, is shown in figure 3.6. During training, we must go through four stages: learning sets composition, feature extraction, intermediary classification and final classification. Each stage has parameters that must be determined. In a training instance, part of these parameters is manually fixed, while the other part is determined automatically. The result of training is a full classification system, which however might not be optimal due to the choices for the fixed parameters.

During validation, the systems resulting from multiple training instances, with different parameter choices, are evaluated, and the one with the best performance is selected. In this way, validation allows us to determine the parameters that are not automatically computed in training. The selected system is then submitted to a final evaluation in the test step.

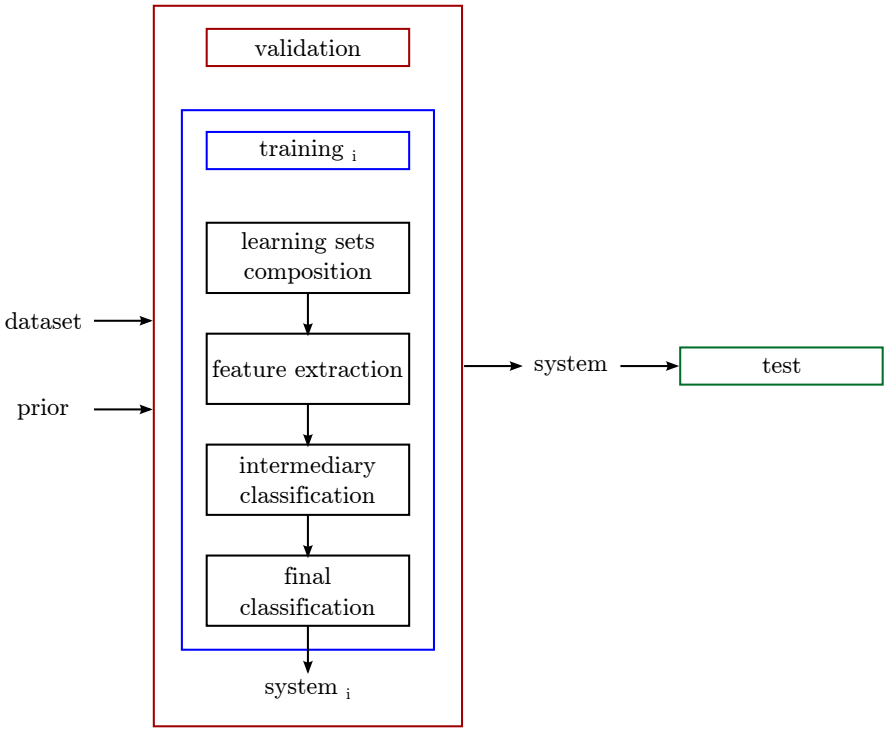


Figure 3.6: *Learning.* Multiple training instances i are launched, each one resulting in a classification system i. These systems are evaluated through validation and one is selected for test. The inputs are the dataset and the prior, while the outputs are the selected system together with its test evaluation.

Overall, the learning inputs are the dataset and the prior information. The dataset is the source of the actual training, validation and test sets chosen during the learning sets composition. The prior corresponds to any assumption, hypothesis or choice made about the system. The features and the classification model, for example, are part of the prior, as well as the set of different parameters used in validation. Seeing it through this point of view, the dataset itself could actually be considered as part of the prior, since it implies which type of environment and sensor setup are being targeted.

The learning outputs are the definitive classification system and its evaluation through the test step. The system is predictive, able to classify new data, and must therefore be able to achieve a certain degree of generalization. In our work, we aim at achieving a basic level of generalization which we call the dataset level. Generalizing at the dataset level means that the system is capable of classifying data coming from a similar environment and acquired with a similar sensor setup. Achieving higher levels of generalization would mean changing the environment or changing the sensor setup.

In the following, we describe the learning process for each of the four stages mentioned above. Chapter 4 describes then how learning is implemented and used to evaluate our approach.

3.4.1 Learning Sets Composition

Learning starts with the choice of the training, validation and test sets. The training set provides the core input data. This data is not only used to train the GMM, but also to find the standardizing parameters and to determine the grouping. A constant goal in any learning task is to be as efficient as possible in the training. This means maximizing the performance of the resulting system, while minimizing the size of the training set. Minimizing the training set's size is attractive because it means that less data needs to be acquired and that less time is spent in training. Overall, it leads to a faster, and probably simpler, learning process.

On the other side, the more information is given to a system, the more the system is likely to perform better. However, this relation depends on the complexity of the classification system. The complexity is determined by many factors, from feature extraction until the classification. Concerning feature extraction, it is affected, for instance, by the dimensionality of the feature space. Detecting patterns in a feature space of higher-dimensionality requires more observations. Concerning the classification, it is affected by the number of components when using a mixture model, as in our case, for example. A model with more components is capable of modelling more patterns, but only if the sufficient number of observations is available.

Concerning the relation between the amount of training data and the performance of

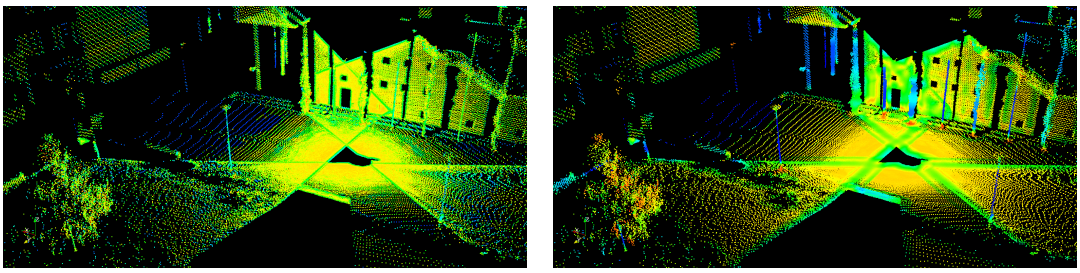
a system, there are two important problems that may apply: underfitting and overfitting. Underfitting happens when a system is not complex enough. This low complexity is the factor limiting the performance. In this case, increasing the amount of training data does not help. Overfitting happens when a system is too complex. This high complexity is prejudicial for the performance because what the system learns is too close to what is present in the training data, and thus the system generalizes poorly. Thus, here the amount of training data is the limiting factor, and increasing this amount can improve the system's performance.

It is possible to characterize this behaviour by analysing the learning curve. Given a certain system, with a certain complexity, we perform training multiple times, each time varying the size of the training set. The performance of each system is evaluated, and a curve of the performances in function of the training set size is built. Such curve is the learning curve. It likely indicates the point from which increasing the data does not bring a suitable increase in performance, because the complexity of the system is limiting its capacity to learn more complex patterns.

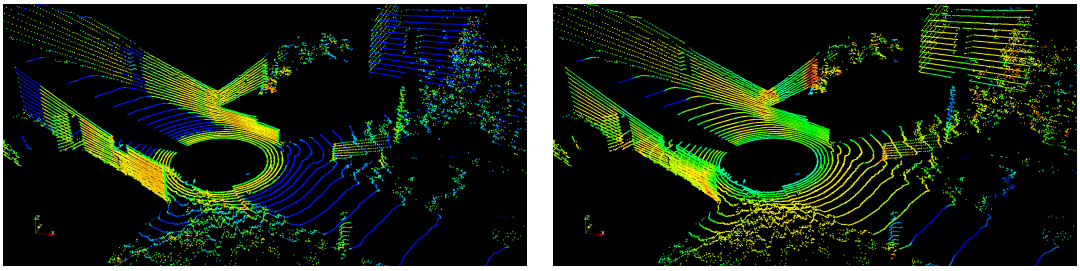
The validation set is also an essential part of learning. As the name suggests, this set provides the data which serves to validate, to confirm the performance of the trained system. For this reason, this set must contain data different from the one in the training set. This is necessary to ensure that the classification system is able to generalize. Here, the ideal is to have a validation set as large as possible. The larger the set is, the more reliable is the evaluation of the system.

The test set has a purpose similar to that of the validation set. It is used to perform a final test on the system selected in the validation step. Analogously to the validation set, this set should be as large as possible. The reason is the same: testing as much cases as possible in order to ensure that the system is capable of generalizing. Because the test set is supposed to be used once, as a final step in learning, normally it is unique. This is not always the case when considering the training and validation sets, because these might change depending on the training and validation scheme adopted.

It should be noted that, when speaking of the size of a set, it is implied that quality is being considered too. Increasing the size means adding data constituting new test cases to the set. In the context of classification, it is also important to try having a balanced quantity of observations from each of the considered classes in a set, in order to ensure that each class is being trained and evaluated on the same terms. This is usually difficult to achieve, however. For this reason, evaluation methods should also take this point into account, as it will be discussed in chapter 4.



(a) x_1 in a Freiburg scan. The columns have a stronger blue colour at the right because their shapes approach the 1D case. Analogously, the base of the columns and the foliage in the tree have a stronger red colour, because their shapes approach the 3D case. Thus, these objects are better distinguished at the larger scale. On the other side, the regions on the ground and on the wall which are at the interface of different sampling densities are coloured differently from the rest of the ground or wall, respectively, and this effect is more pronounced at the larger scale.



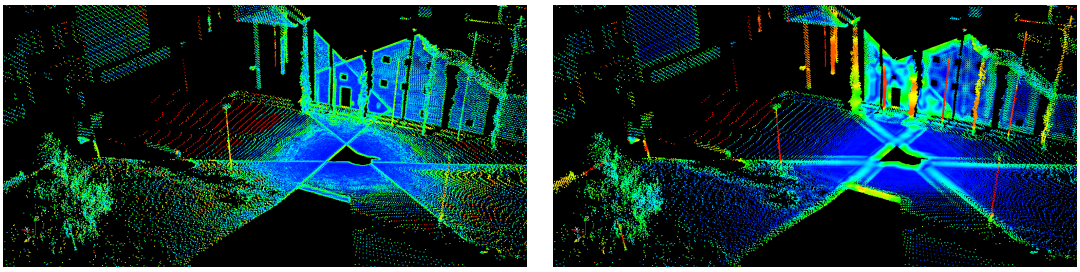
(b) x_1 in a Caylus scan. The ground has a larger surface coloured in blue at the left, meaning that it appears as 1D due to under-sampling. At the larger scale, this effect is diminished. On the other side, the borders of the objects are less clearly perceived at the larger scale, a problem indicated by the pronounced change in their colouring.

Figure 3.7: *Feature x_1 at Different Radii.* $r = 0.2m$ at the left, $r = 1m$ at the right. Colours range from red, indicating the lowest value, to blue, indicating the highest value. The lowest value indicates a 3D shape, while the highest value indicates a 1D shape.

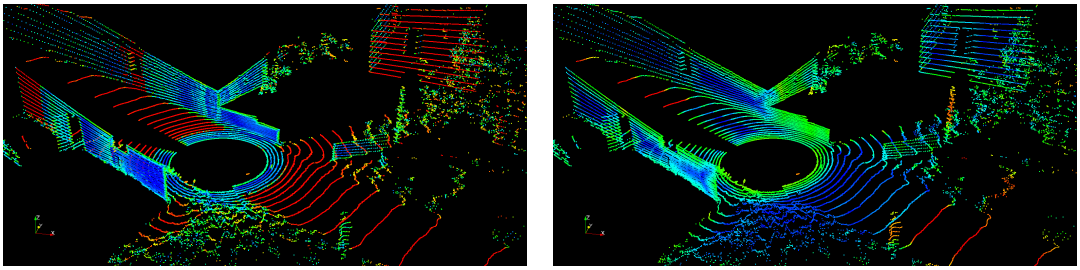
3.4.2 Feature Extraction

The feature extraction stage requires the choice of r or R, the radius or set of radii of the support regions, respectively, as explained in section 3.1. These parameters determine the scales at which the model operates. There are two factors which are central for the choice of the appropriate radii: the capacity of distinction offered at a certain scale, and the relation of the scale with the sampling densities found in a point cloud. These factors are explained below, and illustrated in figures 3.7 and 3.8.

The first factor is how well objects are distinguished at the given scale. At a large scale, isolated objects are better distinguished. Consider, for instance, the case of distinguishing a wall from a post. At a small scale, a region from the post is not that different from a



(a) x_2 in a Freiburg scan.



(b) x_2 in a Caylus scan.

Figure 3.8: *Feature x_2 at Different Radii.* $r = 0.2m$ at the left, $r = 1m$ at the right. Colours range from red, indicating the lowest value, to blue, indicating the highest value. The lowest value indicates a 1D shape, while the highest value indicates a 2D shape. Here, conclusions analogous to the ones made in figure 3.7 apply, but concerning the distinction between 1D and 2D shapes.

region of the wall. At a larger scale, however, the region of the post appears relatively more linear, while the region of the wall appears relatively more planar. In a given environment, there will be a certain minimum scale at which most of the objects can be correctly distinguished.

On the other side, at a large scale, the problem of clutter exerts a more serious influence. If the post were near a wall, the two would appear mixed, and thus would not be correctly perceived. Different environments likely present different amount of clutter, therefore imposing different upper limits on the scale. Thus, the first trade-off that must be dealt with when choosing the radius is how well isolated objects are perceived versus how poorly cluttered objects are perceived.

The second factor influencing the choice of the radii is the interplay of the scale with the different sampling densities found in a point cloud. A sparsely sampled region might not be properly perceived at a small scale, in which case we say the region is under-sampled. For instance, points on the ground or on a wall, which should appear as planar, might appear as linear because not enough neighbours were captured in the support region.

At the other extreme, at a large scale, there is an increased risk that a region of a single object will be sampled at different rates. For instance, if there is a pronounced change in density in some region of a wall, it may appear as linear due to the higher concentration of points in the densely sampled part. This also affects the borders of objects: the border of a wall, because of the empty space next to it, is perceived as relatively linear, instead of planar. This effect is present at smaller scales too, but then the concerned regions are smaller. Thus, the second trade-off involved in the choice of the radii is how well under-sampled regions are perceived versus how poorly regions with different sampling densities are perceived.

The training step is also where the standardizing parameters are computed. Once the features have been extracted for all the points in the training set, the mean and the variance along each feature dimension are computed. These values are kept for use during subsequent feature extraction operations. The training points are standardized using the values just computed, constituting the definitive version of the training set. Other standardizing methods exist, but we choose to use this one because its statistical nature is consistent with the GMM nature.

As discussed above, the parameters of feature extraction left open for manual setting are the radii. Feature selection consists thus in searching for the most appropriate values for the radii. In our approach, feature selection is included in the validation step. The different radii values to be tested are included in different systems to be compared in the validation. In this way, the features are implicitly tested, together with the whole system, based on the unified evaluation performed during validation.

3.4.3 Intermediary Classification

The parameters of the GMM are determined in the training step. The parameter set is denoted by $\theta = \{\pi_1,\ ...,\ \pi_{N_{CY}},\ \boldsymbol{\mu}_1,\ ...,\ \boldsymbol{\mu}_{N_{CY}},\ \Sigma_1,\ ...,\ \Sigma_{N_{CY}}\}$. The training set is denoted by $X = \{\boldsymbol{x}_1,\ ...,\ \boldsymbol{x}_{N_{TS}}\}$, N_{TS} being the number of elements in the set. The set of the latent variables corresponding to the points in the training set is denoted by $Y = \{\boldsymbol{y}_1,\ ...,\ \boldsymbol{y}_{N_{TS}}\}$. From these, only X is given. Finding a function of θ given X that could be optimized should provide a way to determine θ. One approach is to select, for such function, the log likelihood of the data given the parameters, $\ln p(X|\theta)$, which should be maximized. This method is called maximum likelihood [Bilmes, 1997; Bishop, 2006; Hastie et al., 2013]. The problem can be formulated as finding the solution to

$$\text{argmax}_\theta\ L(\theta|X) = \text{argmax}_\theta\ \ln p(X|\theta). \tag{3.8}$$

Taking equation 3.5 into account, the log likelihood can be developed as

$$\begin{aligned}\ln p(X|\theta) &= \ln \prod_{i=1}^{N_{TS}} p(\boldsymbol{x}_i|\theta) \\ &= \ln \prod_{i=1}^{N_{TS}} \sum_{j=1}^{N_{CY}} \pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \Sigma_j) = \sum_{i=1}^{N_{TS}} \ln \sum_{j=1}^{N_{CY}} \pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \Sigma_j).\end{aligned}$$

Let $N_j = \sum_{i=1}^{N_{TS}} p(y_{ij}|\boldsymbol{x}_i)$. The solution of the maximization with respect to each π_j, $\boldsymbol{\mu}_j$ and Σ_j is:

$$\pi_j = \frac{N_j}{N_{TS}}, \tag{3.9}$$

$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{i=1}^{N_{TS}} p(y_{ij}|\boldsymbol{x}_i)\boldsymbol{x}_i, \tag{3.10}$$

$$\Sigma_j = \frac{1}{N_j} \sum_{i=1}^{N_{TS}} p(y_{ij}|\boldsymbol{x}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T. \tag{3.11}$$

It is interesting to note the parallel between this solution and the maximum likelihood solution (also used in equation 3.1) for the estimation of a single Gaussian distribution from the same data:

$$\boldsymbol{\mu} = \frac{1}{N_{TS}} \sum_{i=1}^{N_{TS}} \boldsymbol{x}_i, \quad \Sigma = \frac{1}{N_{TS}} \sum_{i=1}^{N_{TS}} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$$

The difference is that, for the mixture case, each data point is weighted by the class posterior probability, $p(y_{ij}|\boldsymbol{x}_i)$, and the terms are normalized by the sum of the posteriors, N_j, which can be seen as the probabilistic number of points assigned to a component. In the same way, if Y were given, we could estimate each Gaussian component separately,

taking into account only the points associated to the component, and thus using the equations for the single Gaussian case.

The solution for θ is not a closed-form expression, because it depends on the class posterior probabilities. These, in turn, require knowing θ, since we can think of $p(y_{ij}|\boldsymbol{x}_i)$ as $p(y_{ij}|\boldsymbol{x}_i, \theta)$. However, neither of them is given. This situation can be solved by the iterative algorithm of Expectation-Maximization (EM). EM consists, in this case, in

- (Expectation - E step) computing $p(y_{ij}|\boldsymbol{x}_i, \tilde{\theta})$ from a given estimate of the parameters $\tilde{\theta}$, then
- (Maximization - M step) computing a new θ using the equations 3.9, 3.10 and 3.11, then
- if a given convergence criterion is not matched, assign θ to $\tilde{\theta}$, and repeat the procedure.

The algorithm starts with an estimate $\tilde{\theta}_0$. The training of the GMM is thus performed by applying the EM algorithm, as discussed. In our work, it is implemented with *scikit-learn* [Pedregosa et al., 2011].

N_{CY}, the number of classes in the GMM, or the number of intermediary classes, determine how fine is the model with respect to the patterns that it can represent. By increasing the number of classes, the number of patterns is increased. Indeed, with an unbounded number of classes, it is ideally possible to model arbitrary decision boundaries in the feature space. N_{CY} should be large enough so that the GMM is able to provide a fine enough model of the patterns in the environment. Under this condition, we ensure that the corresponding intermediary classes can be grouped afterwards into meaningful final classes.

The number of classes indicate the complexity of the GMM. Increasing this number implies that the EM-based training will be slower, and that a larger amount of training data will be needed. Moreover, and perhaps most importantly, the grouping stage will be made slower, because the expert will have to look at and examine more classes. Ensuring that the grouping process remains simple requires that the number of classes should be kept at a minimum. Therefore, the trade-off faced in the selection of N_{CY} is providing a fine enough representation versus having a simple enough EM and grouping processes.

The initialization is also an essential part of the EM training. EM is guaranteed to converge to a local maximum only, and the quality of the local maximum achieved is determined by the initial parameters. A commonly-used method is to randomly pick points from the training set and use them as the initial means. The probability of sampling points from denser regions is higher, which is advantageous because these regions are probably

the most relevant. The covariances are initialized, equally, with the sample covariance of the whole set. The coefficients are all set to 1 divided by the number of components.

Another commonly-used method is to start with randomly sampled points, in the way just described, and then proceed with a preliminary K-means clustering. The obtained cluster centers are then set as the initial means for the EM, while the remaining EM parameters are set as previously explained. K-means speeds-up the whole process, because it converges faster than EM.

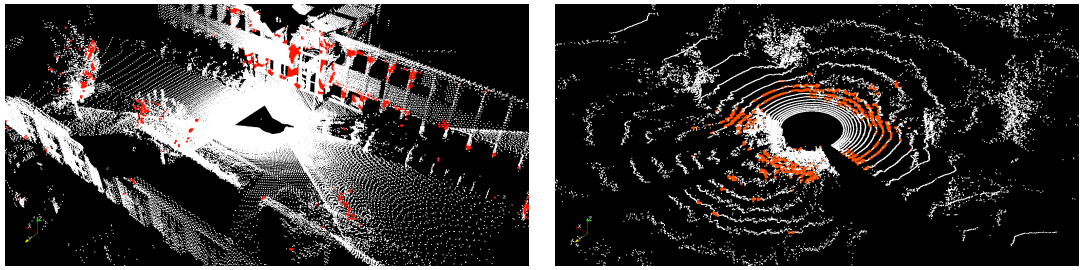
The process of randomly sampling points and applying K-means may be replaced by the improved K-means++ algorithm [Arthur and Vassilvitskii, 2007], which makes an essential alteration: after each cluster center is sampled from the training set, the points are weighted proportionally to their squared distance to the nearest existing center, and then the next center is sampled accordingly. This leads to cluster centers which are more spread between themselves, better exploring the points of the training set. This method is the one employed by default in *scikit-learn*. In fact, here the algorithm is ran 10 times, and the best run is selected in the end. All the methods discussed so far are random. However, this one, due to the combination of the K-means++ algorithm and multiple runs, is the one which produces the least random initialization.

With a random initialization, each EM training results in a different GMM model. This could be considered as a noise in the GMM training. This is a non-desired property, especially when the subsequent grouping is considered, since it is not possible to perform an arbitrary number of groupings in order to select the best one. Thus, we keep the method used in *scikit-learn*. The random nature of the GMM training, even if considerably reduced by the initialization, is a limitation of the approach.

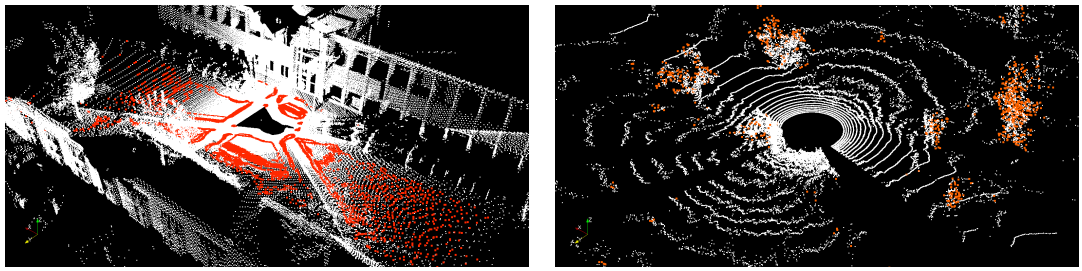
The search for the right number of intermediary classes constitute the model selection. As for feature selection, model selection is performed implicitly in the validation step. Multiple models, with different number of classes, are trained and compared during validation, thus leading to a unified evaluation of the different values.

3.4.4 Final Classification

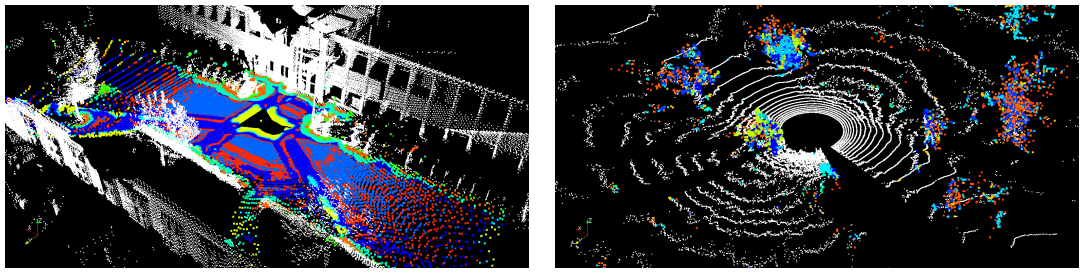
The grouping is determined during training. This step is done in a supervised manner, by a human expert. Overall, it consists in examining the results of the intermediary classification, by visual inspection, and assigning to each intermediary class a final class, or the class *unknown*. This examination is performed on a grouping set, which does not have to be the same as the training set, although it usually is. To perform this task, a graphical interface is required. In our work, we found that the visualization tool ParaView [Moreland, 2013] provided all the desired functions. Some examples of the grouping training are shown in figure 3.9.



(a) Single intermediary classes that fail to represent a single element, and thus might not be grouped, depending on the target task. In the Freiburg case, it could represent either building or vegetation. In the Caylus case, it could represent either road or grass.



(b) Single intermediary classes that represent a single element, and thus are likely to be grouped. In the Freiburg case, it represents ground. In the Caylus case, it represents vegetation.



(c) Intermediary classes that are grouped into one final class. In the Freiburg case, they represent ground, while in the Caylus case, they represent vegetation.

Figure 3.9: *Grouping Training.* These are examples of the actual interface used in the training process. Left: a scan from the Freiburg dataset. Right: one from the Caylus dataset. The concerned classes are highlighted in colours.

The expert must have in mind the target task for which the classification system is being designed. This means selecting a set of final classes with relevant properties. It is implied, however, that these classes are properly distinguished among the intermediary classes. If this is the case, and the intermediary model is indeed able to provide a flexible enough set of classes, then the semantic complexity of the final model can be chosen. For instance, the final class *ground* may be selected, or it may be divided into the more specific classes *road* and *grass*. The semantic complexity required in the target task is thus manually handled by the expert during training.

An important property that may be influenced by the expert in this stage is the precision-recall balance. These metrics will be explained in more details in chapter 4. Here, we may say that when inspecting an intermediary class cy_i, changes in the precision and recall of a final class cz_j can be roughly estimated, visually. If cy_i is grouped under cz_j, the cz_j's precision decreases by the percentage of its points covered by the false positives in cy_i. Analogously, the cz_j's recall is increased by the percentage of its points covered by the true positives in cy_i. Among these two, the recall is easier to visually estimate.

The preceding comments suggest a criterion for the grouping, which is the maximization of the recall. The criterion can be presented as follows: an intermediary class should be grouped under the final class whose recall will have the largest increase, or, if this cannot be clearly determined, marked as *unknown*. This criterion would be advantageous if a post-processing step consisting of a filtering or smoothing operation is applied, in which case a better recall is desired. The actual criterion used, however, would depend on the target task.

The visual nature of the criterion eventually leads to errors by the expert, that is, the expert may take the wrong decision regarding the grouping of an intermediary class. This problem is reduced by the clarity requirement present in the criterion, but never completely avoided. This could be considered as a kind of noise in the grouping training. This noise, together with the noise in the GMM training, induced by the randomness of the EM initialization, constitute the training noise.

Chapter 4

Evaluation

The evaluation of the proposed approach follows the training, validation and test steps, as described in chapter 3. We evaluate the system under two separate contexts, each one corresponding to a different dataset. Both datasets contain 3D point clouds of outdoor environments. The first one is the Freiburg public dataset [Steder et al., 2011], for which we have ground-truth, made available in [Behley et al., 2012]. The second one is a dataset acquired with our own robot and sensor setup, for which there is no grount-truth available. These datasets are described more carefully further in this chapter.

In each case, we train multiple systems to be compared through validation. A complete validation is a search problem, and would consist of training all the possible combinations of parameters. This leads to a combinatorial problem. In a supervised learning context, it might be possible to perform a relatively complete validation, since once the different parameters are chosen, the remaining of the process can continue automatically. This condition holds if the time required is not prohibitive.

In our approach, each training case includes the supervised grouping process. Due to the time required in the grouping and the combinatorial factor of the validation, it is not possible for us to proceed in an exhaustive manner. Thus, we choose to perform a constrained validation, selecting a set of training cases considered as most informative. Concretely, this means testing each parameter at a time, by varying it while fixing the others at relevant values. In the end, this leads to a system which is the best locally, under the selected parameter set, yet it still leads to an informative exploration of the different alternatives. The selection of the parameters is done based on a preliminary training evaluation, which does not include any data from the validation or the test sets. The actual selected parameters will be presented throughout this chapter, in the pertinent sections.

The evaluation metrics are explained in section 4.1. The datasets are described in section 4.2. Section 4.3 evaluates the learning sets composition, section 4.4 evaluates the

	unk	cz_1	...	cz_i	...	$cz_{N_{CZ}}$	rec	$\boldsymbol{F_1}$
cz_1				$fp_{i,1}$				
$\vdots$				$\vdots$				
cz_i	$fn_{i,unk}$	$fn_{i,1}$	...	tp_i	...	$fn_{i,N_{CZ}}$	rec_i	$\boldsymbol{F_{1_i}}$
$\vdots$				$\vdots$				
$cz_{N_{CZ}}$				$fp_{i,N_{CZ}}$				
pre	-			pre_i			-	$\boldsymbol{F_{1_{total}}}$

Table 4.1: *Evaluation Metrics.* This table shows the confusion matrix, with added information about the precision, recall and F_1 scores. The rows indicate the true classes, while the columns indicate the predicted classes. unk refers to the *unknown* final class, cz_i to the i-th final class, pre to precision, and rec to recall. tp indicates the true positives, fp the false positives, and fn the false negatives.

feature extraction, section 4.5 the intermediary classification, and section 4.6, the final classification. Lastly, section 4.7 presents the test results.

4.1 Metrics

The Freiburg dataset contains ground-truth data, therefore allowing a quantitative evaluation. We choose to use the precision, recall and F_1 metrics, as shown in table 4.1. These metrics take into account the classwise performance, which is necessary when dealing with unbalanced data, as is the case in semantic modelling. Moreover, F_1 produces a generic evaluation because it includes both precision and recall. For certain target tasks, it might be desirable to prioritize either precision or recall, and then other metrics can be used. Using the notation of table 4.1, the classwise scores are computed as

$$pre_i = \frac{tp_i}{tp_i + \sum_{j \neq i} fp_{i,j}}, \quad rec_i = \frac{tp_i}{tp_i + \sum_{j \neq i} fn_{i,j}}, \quad F_{1_i} = \frac{2 \cdot pre_i \cdot rec_i}{pre_i + rec_i}, \tag{4.1}$$

while the total F_1 score is computed as

$$F_{1_{total}} = \frac{1}{N_{CZ}} \sum_i F_{1_i}. \tag{4.2}$$

Note how precision and recall are obtained from the confusion matrix by a normalization along a column or row, respectively. These scores, as the F_1, may be averaged to produce a total precision or recall score.

The accuracy metric is also reported. Accuracy is the ratio of the classification hits

over the total number of classified points, and is given by

$$accuracy = \frac{\sum_i tp_i}{\sum_j (tp_j + \sum_{k \neq j} fp_{j,k})}, \text{ or equivalently, } = \frac{\sum_i tp_i}{\sum_j (tp_j + \sum_{k \neq j} fn_{j,k})}. \quad (4.3)$$

It is the sum of the terms in the diagonal of the confusion matrix, divided by the sum of all the terms of the matrix. However, it should only be used as a reference for comparison. Point clouds usually present a relatively high number of points at close distance, and this points usually correspond to the ground, or to a road, resulting in an accuracy with a high bias towards these classes.

For the Caylus dataset, there is no associated ground-truth. Actually, this is an example of a dataset for which ground-truth is difficult to produce, due to two factors: the sampling sparsity and the presence of more natural, non-structured elements. The evaluation, in this case, is done only in a qualitative manner, by visual inspection. This case represents what would be a real application of our system: starting from a dataset with no ground-truth, and ending with a visual inspection of the classification results.

It would be possible to base the visual inspection on the precision and recall metrics, by visually examining the classes' hits and errors and making estimates, but this would be too cumbersome and prone to error. The great variance in the point density in function of the distance is enough to make such estimates not reliable.

Alternatively, we note that not the absolute, but the relative difference in recall between scans can be noticed. It is, most of the times, clear enough to see missing points in one scan, compared to another. Therefore, the evaluation is done in a relative way. Scans are compared between them, and differences on the recalls are noted down. This allows a ranking to be established. This procedure is consistent with the grouping method adopted, which also prioritizes the recall. It should be noted that, in fact, the actual criterion used in the visual evaluation would depend on the target task, and could possibly prioritize different factors, other than the recall.

4.2 Datasets

4.2.1 Freiburg Dataset

This is a public dataset, acquired at the Freiburg University's campus [Steder et al., 2011]. There are 77 scans, each scan containing from 150,000 to 200,000 3D points. It contains artificial elements such as streets, buildings of different types, road signs and lamp posts, but also some natural elements such as trees of different shapes and sizes, shrubs and vegetation areas. Some people appear in the scans too.

The dataset was acquired with a SICK LMS lidar, moved using a pan-tilt unit, on a mobile robot. The SICK LMS sensors [SICK, 2015a,b] are 2D lidars, that is, they scan

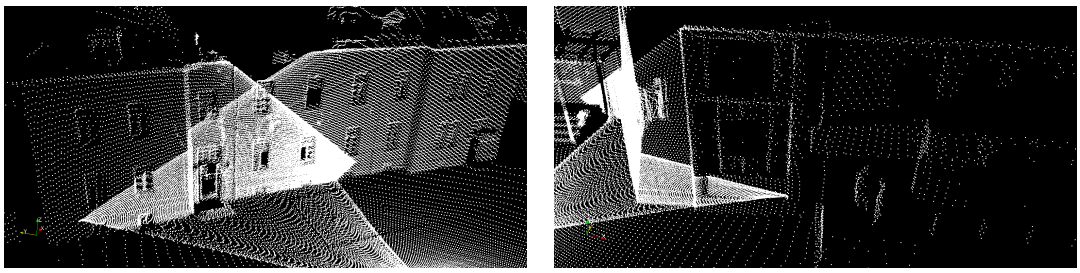


Figure 4.1: *Freiburg Dataset.* The image at the left shows a region where the individual scans overlap, causing important changes in the point densities. The image at the right shows a facade with a variety of window types.

along a plane. There is one laser emitter, and through a rotating mirror, each emission is redirected to a different orientation on the plane. The angular resolution can be set to 0.25°, 0.5° or 1°, and the field-of-view covers 180°. The maximum operating distance is 80m. These sensors use the time-of-flight method for computing distances. The pan-tilt unit is used to change the pitch orientation of the sensor, allowing a scanning along multiple planes, and resulting thus in a 3D scan.

The acquisition was static: the laser acquired the points while the robot was stopped. At each location, three scans at different orientations were taken and merged together. The individual scans overlap each other, creating different sampling densities at the overlapping regions. The scans are expressed in the robot's reference frame, and are accompanied by the respective robot's positions. The robot frame is taken to be at ground height. We assume the sensor is located 1m above it, and thus are able to determine the sensor-to-world transformation.

The Freiburg environment is relatively flat, structured and uncluttered. The point clouds are relatively dense. There are two main challenges encountered in the data. The first one is the nonuniform sampling, consisting of significative changes in the sampling density at the overlapping areas. The second one is the presence of some complex facade features, such as windows, doors, roof and prominent features in general, all of them in varied sizes and types. Such complex features are one example of the environment variability problem. Figure 4.1 shows some examples of these.

For this dataset, the set of final classes is composed by four classes. They were not manually pre-selected, but instead, discovered on the preliminary training evaluation. They were checked against the ground-truth available, to ensure that the latter could be used to support the evaluation. The classes are the following:

- *ground.* It corresponds to road, lawn, sidewalks, and so on. Geometrically, these are flat and planar, with normals oriented upwards.

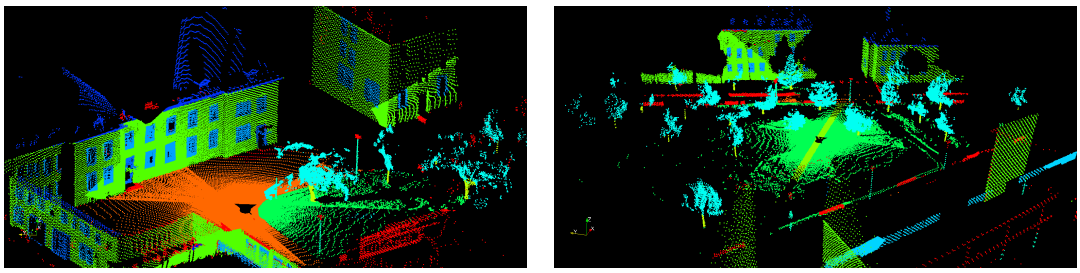


Figure 4.2: *Freiburg Ground-truth.* The colours encode the class labels. In the first picture, at the bottom-right, a big structure, probably built of glass due to the presence of multiple points inside it, is marked in red, indicating that it is not labelled. In the second picture, some non-labelled regions in red can also be seen.

- *building.* It corresponds to buildings, including any facade structure, roofs, and so on, and also to shrubs. Shrubs are included here because they are so precisely trimmed that they appear clearly as low walls. Only in scarce cases, some edges present random traces indicating vegetation. Geometrically, the facades and shrubs are planar with normals oriented along the horizontal plane, whereas the more complex structures have more varied geometries.

- *post.* It corresponds to posts, tree trunks and people. Geometrically, these are linear, with normals oriented along the horizontal plane.

- *vegetation.* It corresponds to vegetation, tree foliage and to bicycles and bicycle stations too. Parked bicyles and bicyles stations are common in the dataset, and their relatively random and scatteered shape matches well that of vegetation in general, therefore being included here. Geometrically, they are scattered, three-dimensional, with normals oriented in unpredictable directions.

The ground-truth presents a fine distinction of elements, with twenty classes in total. These include, for example, *ground*, *sidewalk* and *lawn*, as well as *facade*, *window* and *door*. These are grouped into the smaller set of final classes. We follow approximately the work done in [Behley et al., 2012], for which the ground-truth was produced, and where the classes are also grouped for the evaluation. The classes considered in their case were: *ground*, *facade*, *pole* and *vegetation*. These are relatively similar to ours, except that we include bicycles as vegetation, whereas they leave out the bicycle points from the evaluation, and that we include shrubs as facade, instead of as vegetation. Figure 4.2 shows some examples of the ground-truth.

Another point to be mentioned is that the ground-truth does not cover all the points in the scans. Some complex features are left out, such as glass facades and the roofs of

bicycle stations. Isolated groups of points, and some erroneous artifacts, are also filtered out. These points are thus not used in the evaluation. They are, however, still present in the training set, which means that they still contribute to the training of the model.

4.2.2 Caylus Dataset

This dataset was acquired with our own robot and sensor setup. It contains a few thousand scans, each one with approximately 76,800 points. The scenario is an artificial countryside village. It presents a great variety of natural elements such as low and high grass, trees, bushes and other vegetation, but also artificial elements like an asphalted road, buildings, and some abandoned vehicles. The operator of the robot can be seen in some scans.

The scans were acquired with a Velodyne HDL-32 lidar [Velodyne, 2012], mounted on the top of a Segway RMP-400-based UGV. The Velodyne HDL-32 is a 3D lidar, scanning with a horizontal field-of-view of 360° and a vertical field-of-view of 41.3°. It has 32 emitters that fire along a vertical plane, as illustrated in figure 4.3. It also has a rotating head, which combined with the emitter arrangement, produces a 3D scan. The head rotates at 10Hz. The sensor has a horizontal angular resolution of approximately 0.16° and a vertical angular resolution of approximately 1.33°. The maximum operating distance is 70m. The distance computation method is time-of-flight.

The Velodyne lidars, including the HDL-32, are designed to allow mobile acquisition. The dataset was acquired in this way. The UGV was manually controlled by an operator, while the lidar acquired data and a SLAM method, namely RT-SLAM [Roussillon et al., 2011], provided the localization by fusing GPS, inertial and visual information. Each full revolution of the sensor's head produced a 360° scan, with points being transformed into the sensor's reference frame at the beginning of the revolution.

The area of the dataset presents some gentle slopes at specific points. Otherwise, it is basically flat. It is less structured than the Freiburg area, with more grass, vegetation and some natural terrain. However, the two main challenging characteristics are the nonuniform sampling and the clutter. The nonuniform sampling is a consequence of the sensor's sampling pattern. In a Velodyne scan, at close range, the sampling is relatively dense, but moving to farther ranges, the density decreases very fast, resulting in sparsely sampled regions. The second challenge is the important amount of clutter present in the environment, concerning particularly tree trunks, often surrounded by vegetation. Figure 4.4 shows examples of these phenomena.

For this dataset, after the preliminary training evaluation, the set of final classes was composed by the six discovered classes:

- *road*. It corresponds to the asphalted road, and to sidewalks. Geometrically, these are planar, with normals oriented upwards.

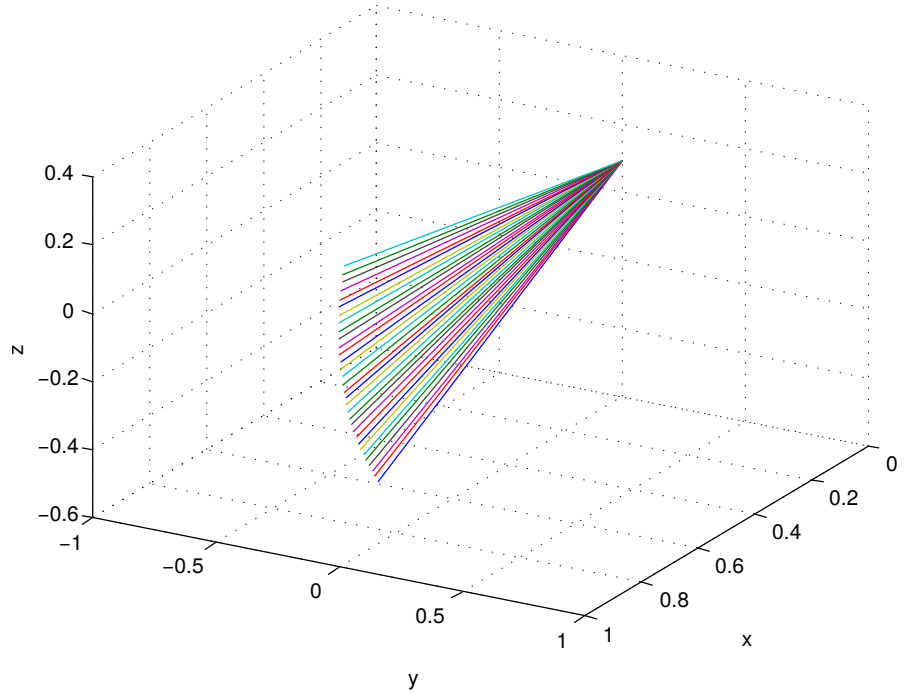


Figure 4.3: *Velodyne HDL-32 Emitter Arrangement.* This was determined on the basis of the calibration system and data provided with the sensor.

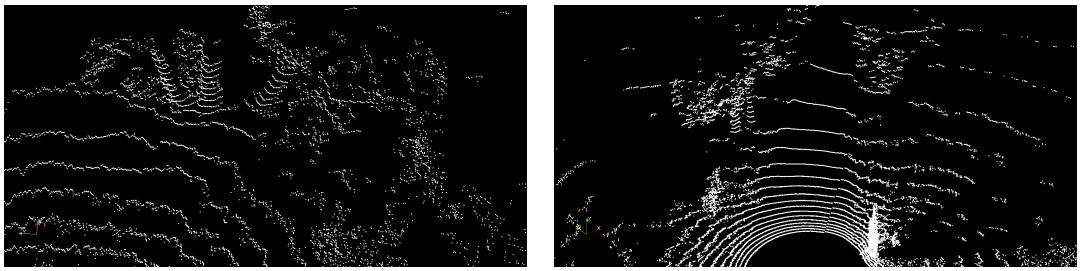


Figure 4.4: *Caylus Dataset.* The image at the left shows an example of clutter found in the set. At the top-left of it, we can see two tree trunks being surrounded by foliage and vegetation. The image at the right shows the sampling sparsity problem. It is particularly noticeable for the road, going from the bottom-center to the top-center of the image, and presenting a dramatic decrease in sampling.

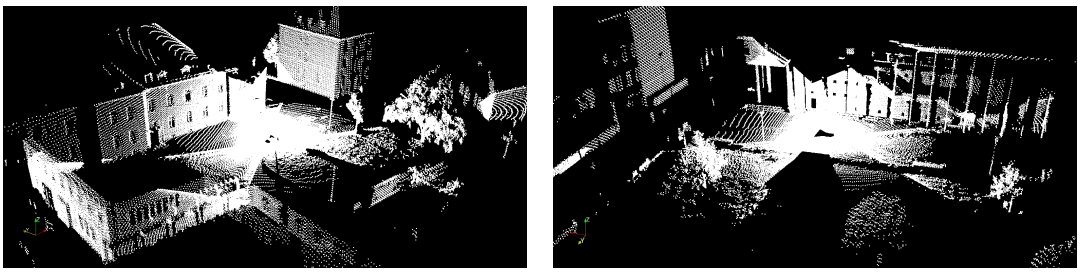
- *building*. It corresponds to buildings and facade features, such as doors and windows. Geometrically, these are planar, with normals oriented along the horizontal plane, except from the facade features which have varied geometries.
- *trunk*. It corresponds to tree trunks, posts and people. Posts are rare, but still present in the dataset. People refers mainly to the robot operator who appears in most of the scans. Geometrically, these are linear, with normals oriented mainly horizontally, but sometimes in other directions, for example when a trunk is inclined.
- *vegetation*. It corresponds to tree foliage and vegetation. Some regions where the grass is high can be considered as vegetation too. Geometrically, these are scattered, three-dimensional.
- *grass*. It corresponds to grass. Geometrically, it is basically planar, but less than road, because of the more scattered pattern of the grass.
- *rough*. It corresponds to rough terrain, usually found at the interface of grass and vegetation, or at the base of trees. It also corresponds to regions of medium-high grass. In fact, the classes *grass*, *rough* and *vegetation* represent a progression of unstructured terrain, and of scatterness, in terms of geometric shape.

4.3 Learning Sets Composition

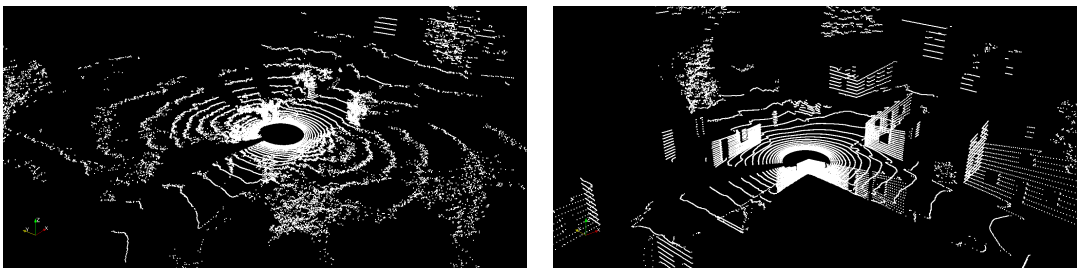
The implementation of the adopted validation method is done by defining one validation and one test sets. The validation set is used to evaluate every training case, while the test set is used to evaluate the selected system. The training set, however, may vary in each case, as explained in chapter 3. Varying the training set allows the establishment of a learning curve, which is the goal of this section.

From each dataset, 10 scans were reserved for use in the different training sets, 5 for the validation set and 5 for the test set. The training scans were the first ones to be chosen, followed by the validation scans, and finally by the test scans. Assigning the priorities in this manner ensures that the system will learn with the best data available. The selected scans were kept as spread as possible over the scenes, while at the same time being picked from the most interesting areas, and aiming at having as much balance as possible between the different elements.

Regarding the training set, one of the goals being the minimization of its size, the decision made was to not use all the data available. Indeed, the datasets have many more scans, so data was not a limiting factor. An examination of the data also revealed that five scans already contained a reasonable amount of the main elements, for both datasets. Lastly, reserving too much scans for training would have a negative impact on the



(a) Freiburg



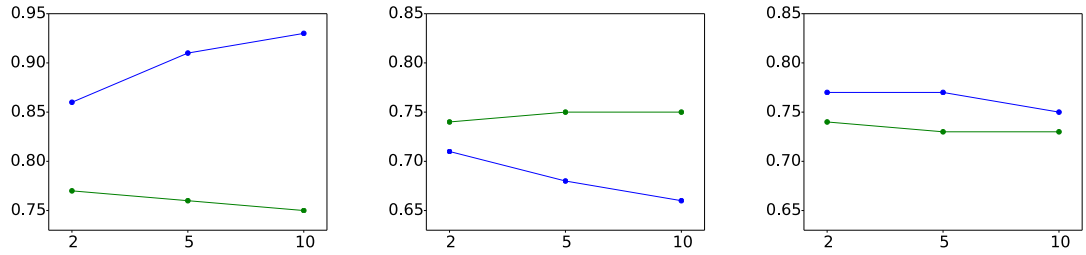
(b) Caylus

Figure 4.5: *Training Sets.* These are the training sets with 2 scans and $r = 0.6m$. The points filtered out at feature extraction were already removed.

composition of the validation and test scans, because there would be less interesting and original scans available. Thus, reserving 10 scans for training seemed a good compromise, as well as using 5 scans for validation and another 5 for test.

Using the scans reserved for the training, three training sets are tested, containing respectively 2, 5 and 10 scans. Figure 4.5 shows the smaller training sets for Freiburg and Caylus. We test two feature configurations: a single-scale one, with $r = 0.6m$, and a multi-scale one, with $R = \{0.2m, 0.4m, 0.6m, 0.8m, 1.0m\}$. This choice of R aims at covering the relevant scales, from the smaller to the bigger ones. r is picked as the middle scale, which should constitute a good compromise. Moreover, these settings provided promising results in the preliminary training evaluation. The actual evaluation of the feature parameters is done in section 4.4.

The number of intermediary classes is fixed at $N_{CY} = 50$. Choosing a larger number of classes would be prohibitive for the grouping training, so this number is considered as the maximum tractable number. As the system's performance is expected to improve proportionally to N_{CY}, the maximum tractable value was chosen. This value was also verified in the preliminary training evaluation. The evaluation regarding the number of intermediary classes is performed in section 4.5. Lastly, concerning the grouping set, it is taken to be the same as the training set with 5 scans.



(a) *precision* (b) *recall* (c) F_1

Figure 4.6: *Freiburg Learning Curves.* The plots show the scores in function of the number of scans in the training set. The scores are the total precision, recall and F_1. The total F_1 is the average of the classwise F_1 scores, so the precision and recall shown are not directly used in the computation, but rather indicators of the global trend across the classes. Blue indicates the single-scale results, green indicates the multi-scale results. The accuracy scores fell between 0.83 and 0.87.

4.3.1 Freiburg Results

Figure 4.6 shows the learning curves for the Freiburg tested systems. The first two single-scale systems obtained $F_1 = 0.77$, being the ones that performed the best. Overall, the performances, indicated by the F_1 score, decrease when the number of scans in the training set is increased. The precision and recall curves reveal the reason: for the single-scale system, the recall decreases, while for the multi-scale system, the precision decreases, which drags the F_1 down.

In the single-scale case, the recall decreases but the precision increases. What could be happening is that adding points to the feature space is not qualitatively changing the distribution, but quantitatively. The points added are reinforcing the core of the patterns, while at the same time filling their periphery, filling the interface between patterns, and creating smaller patterns, therefore making the division boundaries less clear. The GMM components would adapt to this change, the core ones becoming more precise, the remaining ones becoming more scattered, less precise, and thus being discarded in the grouping, lowering the recall. In the multi-scale case, the opposite is observed, although in smaller proportions, as the curves are smoother. It seems, therefore, that the multi-scale feature space provides different properties to the system.

In spite of the overall decrease, the scores are very close, being in the range $[0.75,\ 0.77]$ for the single-scale and in $[0.73,\ 0.74]$ for the multi-scale. The variations observed could be simply a consequence of the noise in the GMM and grouping training. Thus, it is not possible to point a definitive cause for the observed behaviour. However, it is possible

to conclude that the system's performance does not change much across the tested cases. This indicates that the system, regarding its complexity, is already at its full potential with respect to the size of the training set. From the other point of view, it also means that the smaller training set already provides a good representation of the patterns encountered in the environment.

In order to deepen the analysis, and obtain curves that could reveal the behaviour of the system more sharply, it would be necessary to examine more cases, consisting of smaller and bigger training sets. Additionally, a change that could impact the results is to replace the composition of the sets on a scan basis by one on a point basis. In such method, the sets would be composed using the number of points as parameter, and the points would be randomly sampled from the scans reserved for the sets. This would have two positive effects, the first being the reduction of the scans biases in a set, and the second being the use of the number of points as a more practical parameter in the composition.

4.3.2 Caylus Results

For Caylus, the qualitative evaluation allows the establishment of a ranking among the systems. We use the symbol $>$ to denote that a system performed better than another, and is consequently in a higher position in the ranking. We use the symbol $=$ to denote that two systems performed equally well, and are thus equally positioned in the ranking. This notation will be used for all the qualitative evaluations. The evaluation is done independently for the single- and the multi-scale cases. Here, we refer to a training set simply as set. The rankings are the following:

- Single-scale: 10-scan-set $>$ 5-scan-set $>$ 2-scan-set.
- Multi-scale: 10-scan-set $=$ 5-scan-set $>$ 2-scan-set.

Among the single-scale cases, the 2-scan-set system had a lower recall on *grass*, with respect to the two others. The 5-scan-set system, in turn, had a lower recall on *trunk*, with respect to the 10-scan-set system. Among the multi-scale systems, the only difference was a lower recall on *road* and on *grass* for the 2-scan-set system.

Overall, the performances were higher when the training set size was increased. Therefore, in comparison to Freiburg, the system was able to integrate more scans. It is true that a Caylus scan has less than half of the points of a Freiburg scan. However, the number of points alone is not necessarily informative. The important factor guiding the clustering of the GMM is the relative number of points belonging to the different patterns. A scan from Caylus, in this respect, offers less data then a scan from Freiburg.

The fact is, however, that the differences were not very significant, as for Freiburg, and could be due to the training noise. Increasing the size of the sets could lead to better

results, but the small improvements observed suggest that a 10-scan training set provides a good setup for the single-scale system, in the same way as a 5-scan training set, for the multi-scale system. These training sets are good enough for the system complexity. To definitively clarify the results though, the same two changes pointed out for Freiburg could be applied here: testing smaller and bigger sets, as well as using a point-based set composition.

4.4 Feature Extraction

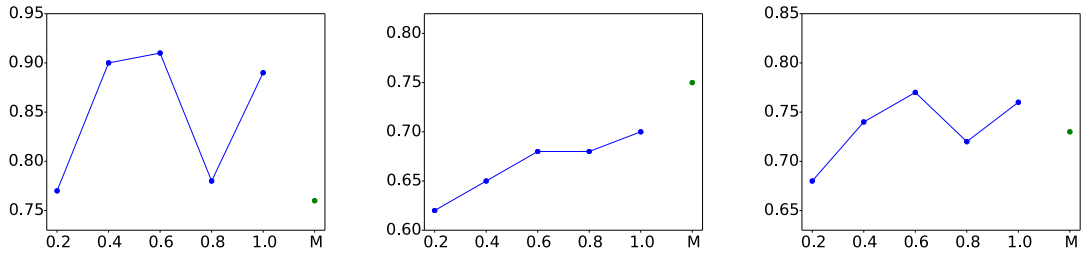
In this section, the feature parameters, r and R, are evaluated. Five different values of r are considered, $0.2m$, $0.4m$, $0.6m$, $0.8m$ and $1.0m$, as well as a multi-scale version with $R = \{0.2m, 0.4m, 0.6m, 0.8m, 1.0m\}$. The training sets used are the ones with 5 scans, since as mentioned before, it was expected that 5 scans allow the necessary elements to be included in the set. The number of intermediary classes is chosen, again, as $N_{CY} = 50$. The percentage of points kept in the feature extraction, that is the points with 4 or more neighbours, is also presented.

4.4.1 Freiburg Results

Figure 4.7 shows the scores computed for the different systems. The system which achieved the best performance was the one with $r = 0.6m$, with a score of 0.77. This shows that this radius was indeed a good choice to be used as basis in the validation. It is closely followed by the one with $r = 1.0m$, with a score of 0.76. Besides this two, and apart from the one with $r = 0.2m$, the scores are not so far away, remaining inside the range $[0.72,\ 0.77]$. The $0.2m$-radius system obtained the worst scores among the single-scale ones, including its classwise F_1 scores, not shown here. The percentage of points kept in the feature extraction, for the different systems, is shown in figure 4.8a.

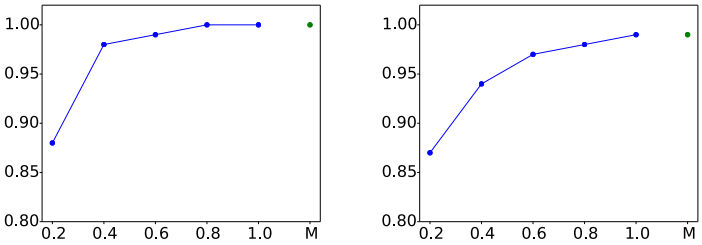
The worse performance of the $0.2m$ system and the overall better performance of the larger-scale systems suggest that bigger radii are more appropriate for the Freiburg scenes. Using bigger radii allows for a better distinction of the elements, and also copes with the sampling sparsity present in the farthest regions of the point clouds. Moreover, the Freiburg environment is relatively uncluttered, so the bigger radii can be used with less negative effects due to clutter.

The multi-scale system obtained the fourth position, with a F_1 of 0.73. Thus, in this case, using all the scales does not lead to the best system. This could be due to the inclusion of non-representative radii, such as the $0.2m$ radius. This radius could introduce a negative influence in the clustering results because, through its corresponding feature dimensions, points will look more similar. Another factor that could limit the performance



(a) *precision* (b) *recall* (c) F_1

Figure 4.7: *Freiburg Radii Results.* The scores are given in function of the radius, noting that the multi-scale system is denoted by the letter "M". Blue indicates the single-scale results, green indicates the multi-scale result. The accuracy scores fell between 0.80 and 0.89.



(a) Freiburg (b) Caylus

Figure 4.8: *Percentage of Points Kept in the Feature Extraction.*

is the inclusion of irrelevant radii. As observed, the larger-scale systems obtained all a relatively similar performance, indicating that some of the larger radii are irrelevant if used together, not adding any new viewpoint.

A clear peak was not present in the result curves, since the systems with $r = 0.6m$ and $r = 1.0$ reached very similar scores, of 0.77 and 0.76, respectively. The exploration of different radii could lead to finer-shaped curves, possibly reinforcing the conclusion that the radius $0.6m$ provides the best performance, and that, overall, bigger radii are more adapted to the environment than the smaller ones. Additionally, it could allow the detection of an upper limit for the radius, above which the performance starts decreasing due to the problems of clutter and density bias. Lastly, testing a different combination of radii for the multi-scale system, possibly with larger and more spread radii, could also prove advantageous.

4.4.2 Caylus Results

The qualitative results for Caylus are the following, denoting the multi-scale system by the letter "M":

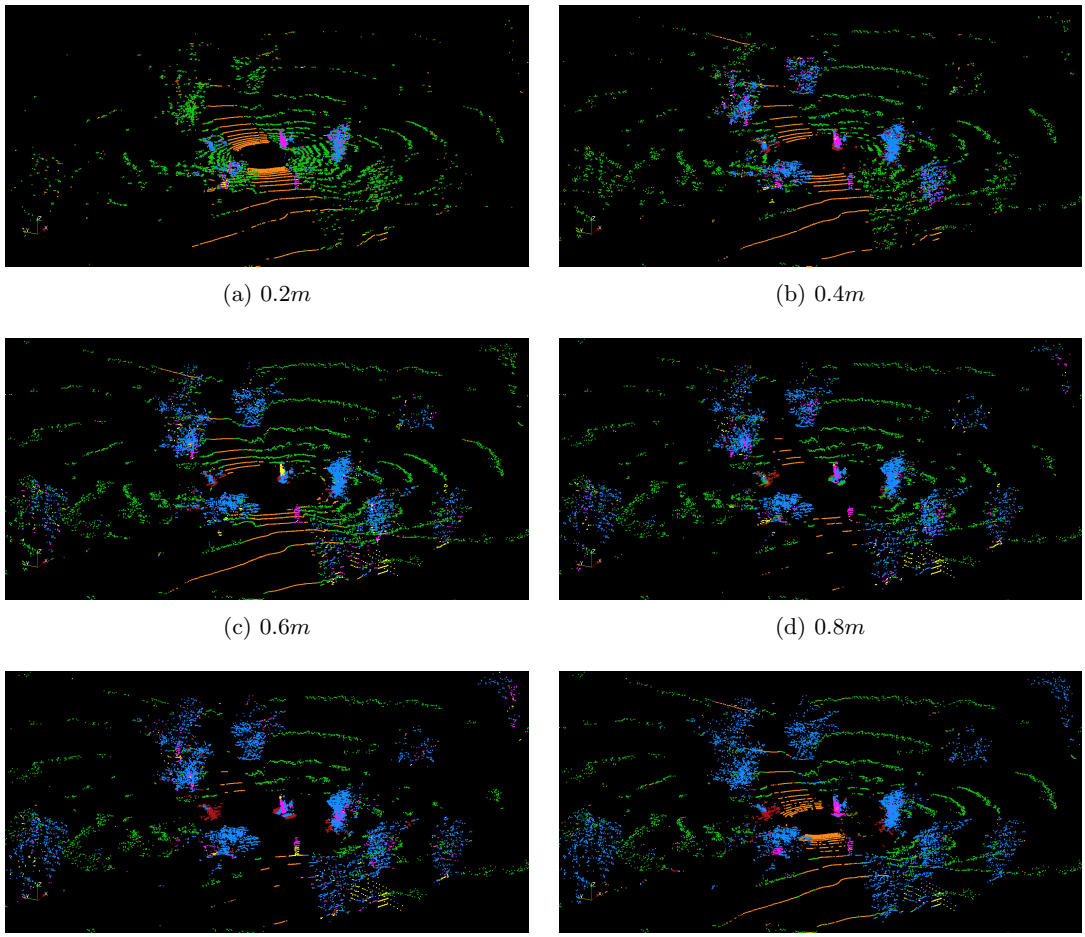
$$\mathrm{M} = 0.6 > 0.2 = 0.4 = 0.8 = 1.0 \ .$$

Figure 4.9 shows the results in one scan from the validation set. Figure 4.8b shows the percentage of points kept in the feature extraction. An interesting pattern appears in the results: the lowest scale offered the best recalls for *road* and *grass* at close range, the largest scale offered the best recalls for *vegetation* and *grass* at far range, and the classes in between followed an apparent graduation from one behaviour to the other. The multi-scale system provided a balanced classification, detecting from the close-range road to the far-range vegetation.

Inspecting the results, the systems with $r = 0.6m$ and multi-scale were elected the ones with the best performances, because they were the most balanced in terms of recall, that is, they achieved a reasonable recall for all classes. Once more, using 0.6 as the base radius revealed itself to be a good choice. Comparing the two best systems, we observe that the single-scale has a lower recall on *road* and *vegetation*, while the multi-scale has a lower recall on *building* and *grass*. Thus, each one has different weak points that cancel out. The other systems have different and complementary weak points too, but they stand at a lower level of performance, in general.

The road and grass at close range are better distinguished at a smaller scale because the characteristic scattered pattern of grass is more apparent under this condition. At a larger scale, on the contrary, the scatterness of the grass is filtered and the grass is then perceived as similar to the road. This effect is also increased by the fact that the grass near the road is frequently shorter, in comparison to more distant grass, being perceived naturally as more planar. Analogously, the grass and vegetation are better distinguished at larger scales, because the difference in the more planar shape of the first and the more three-dimensional shape of the latter becomes more apparent.

Another factor contributing to the distance-dependence of the results is the sampling sparsity of the data. The point clouds in this set present a sharp decrease in sampling density in function of the distance. This causes, for instance, the confusion of *road* with *building* at far distances, because both are perceived just as lines. This characteristic exerts its strongest influence at the lower scales, and is gradually overcome when increasing the scale. But increasing the scale, in turn, brings another effect: the confusion of classes due to clutter, characteristic of the Caylus scenario. This applies, for example, for the class *trunk*, because most trunks are surrounded by vegetation or stand close to other trunks. It also impacts the distinction between *road* and *grass*, making large regions at the interface of both classes be incorrectly perceived.

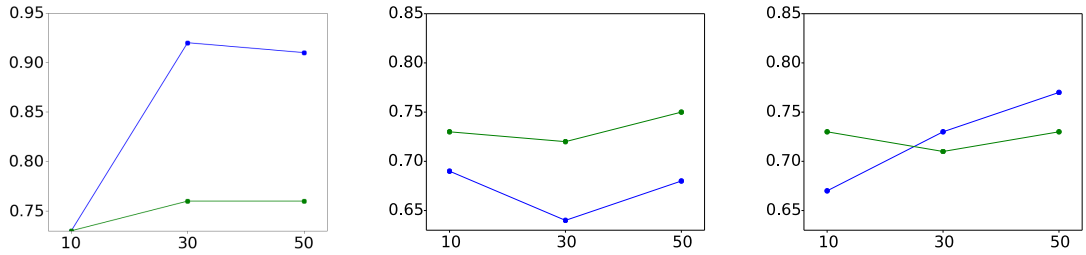


(a) 0.2m (b) 0.4m

(c) 0.6m (d) 0.8m

(e) 1.0m (f) multi-scale

Figure 4.9: *Caylus Radii Results in Scans.* Colours: (*road*, orange), (*building*, yellow), (*trunk*, pink), (*vegetation*, blue), (*grass*, green), (*rough*, brown). Note how the nearby points gradually disappear, and how the correct classification of foliage and grass reaches longer distances, when the radius is increased. The multi-scale system provides a relatively accurate classification at both near and far ranges.



(a) *precision* (b) *recall* (c) F_1

Figure 4.10: *Freiburg N_{CY} Results.* Scores in function of N_{CY}. Blue indicates the single-scale system, green indicates the multi-scale. The accuracy scores fell between 0.81 and 0.87.

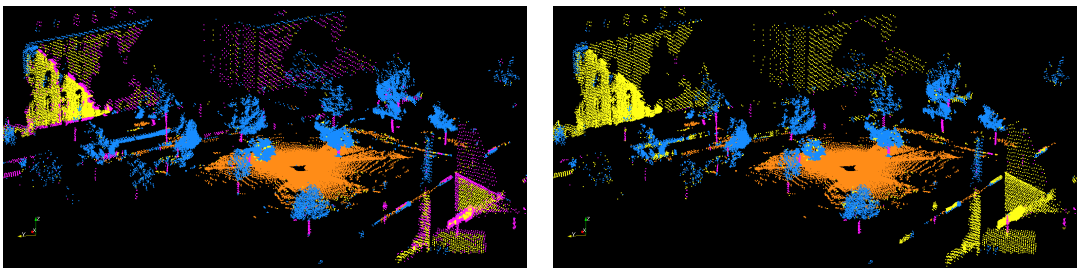
The multi-scale system was able to achieve a balance between the elements at close distance and the ones at far distance. This is possibly due to the efficient combination of the different scales, each one offering better performance for a specific element. This was not the case for Freiburg, where the scales used were possibly not all advantageous, or not complementary enough. In Caylus, because the changes with the scale are more dramatic, as just discussed, the advantages of the multi-scale features become more significant. In fact, comparing these results with the ones of Freiburg, we note that here a curve with a clear global maximum, in addition to the multi-scale system, was obtained.

4.5 Intermediary Classification

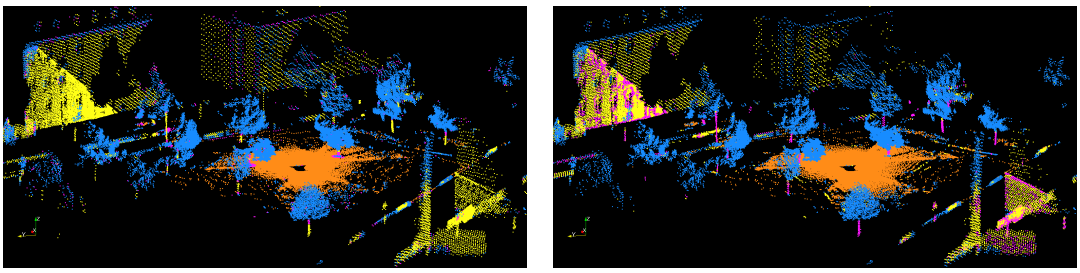
This section evaluates the number of intermediary classes, N_{CY}. As mentioned before, using 50 classes is considered to be the maximum number acceptable, because higher numbers would make the grouping training slower, and therefore too cumbersome. We thus test models with 10, 30 and 50 classes. As before, the training sets used are the ones containing 5 scans, while the feature configurations used are the single-scale one, with $r = 0.6m$, and the multi-scale one, with $R = \{0.2m, 0.4m, 0.6m, 0.8m, 1.0m\}$.

4.5.1 Freiburg Results

The results for different N_{CY} are shown in figure 4.10. Among the single-scale systems, the one with $N_{CY} = 50$ achieved the best score. Among the multi-scale systems, however, the ones with $N_{CY} = 10$ and $N_{CY} = 50$ were the best, reaching the same score. The total precision and recall do not seem to provide any helpful tendency that could clarify the final behaviours.



(a) *Single-scale*. The model with 10 classes misclassified the borders and linear patterns of the buildings as posts, which was the main factor for its worse performance.



(b) *Multi-scale*. The model with 10 classes misclassified most trunks as buildings, resulting in a lower recall for *trunk*. The model with 50 classes misclassified the borders and linear patterns of buildings as trunks, resulting in a lower recall for *building*. In the end, these weak points balanced each other and the final scores were the same for both models.

Figure 4.11: *Freiburg* N_{CY} *Classification Results*. At the left, $N_{CY} = 10$, at the right, $N_{CY} = 50$. Colours: (*ground*, orange), (*building*, yellow), (*post*, pink), (*vegetation*, blue).

The single-scale curve is clearly increasing in proportion to the number of GMM classes, as it was intuitively expected, since with more classes, the GMM should be able to capture more elementary patterns. The multi-scale curve, on the contrary, suggests that the performance may oscillate. A possible explanation is that there could be, indeed, some specific number of classes that generate better clusterings, because the classes would be able to fit better in the different patterns of the feature space. This effect could be more pronounced on the multi-scale system, because of its greater complexity.

In the end, a finer analysis, including more tested cases, would have to be done to verify the extent of the multi-scale case phenomenon. It could be, here again, that the results are simply being affected by the training noise. In fact, this is likely because in the multi-scale the values are quite close. Lastly, figure 4.11 shows some of the classification results for one of the validation scans.

4.5.2 Caylus Results

The Caylus results are the following, noting that the rankings are established for the single- and the multi-scale systems independently:

- Single-scale: $50 = 30 > 10$.
- Multi-scale: $50 > 30 > 10$.

The systems with $N_{CY} = 30$ and 50 achieved the best performances in the single-scale case. The system with $N_{CY} = 50$ achieved the best performance in the multi-scale case. Although no oscillations are present, the differences are still not significant enough to confirm a definitive pattern, especially in the light of what was observed in the Freiburg results. Thus, the same comments made for Freiburg apply here: it could be that the performances are being subject to training noise, and in any case, a finer and more extended analysis is necessary to verify the behaviour of the system.

4.6 Final Classification

In this section, we analyse briefly, and from a global point of view, the supervision complexity resulting from the training of the grouping. The supervision process is simple in comparison to the labelling of a whole dataset. This is due to the availability of the intermediary classification results, which serve as a guide to the supervised grouping. However, in terms of computational complexity, and looking at the system as a whole, we note that the amount of supervision is actually increased. Instead of a single, preliminary supervised labelling, the method requires a supervised grouping for every system submit to the validation step, leading to a linear complexity. Thus, as previously commented, the number of systems that may be explored through validation is constrained by the supervised grouping, whereas in a supervised approach, this number is limited only by the computational resources.

It is difficult to assess whether the various guided, supervised groupings occuring during validation are effectively simpler than a single, not-guided supervised labelling occuring as a preliminary step. The assessement can only be made on a case-by-case basis. If the feature and model selection problems are simple, that is, if the optimal features and model are approximately known, then only a few systems need to go through validation, and then our approach is likely simpler. Otherwise, if the feature and model selection problems are complex and an important number of systems need to go through validation, then our approach is likely in disadvantage.

4.7 Test

The evaluations made in the previous sections constitute the validation stage. Now, we select the best systems and submit them to a last step of evaluation: the test. The base is provided by the test set.

4.7.1 Freiburg Results

From the validation results, we retrieve that the setups which obtained the best performances were:

- regarding the training sets, {2-scan-set, $r = 0.6m$, $N_{CY} = 50$} and {5-scan-set, $r = 0.6m$, $N_{CY} = 50$};
- regarding the feature radii, {5-scan-set, $r = 0.6m$, $N_{CY} = 50$};
- regarding the number of intermediary classes, {5-scan-set, $r = 0.6m$, $N_{CY} = 50$}.

This leaves us with two setups, {2-scan-set, $r = 0.6m$, $N_{CY} = 50$} and {5-scan-set, $r = 0.6m$, $N_{CY} = 50$}. Considering that a smaller training set is better, the best system is thus {2-scan-set, $r = 0.6m$, $N_{CY} = 50$}.

In the set of final classes, {*ground*, *building*, *post*, *vegetation*}, *ground* has the purest semantic interpretation. *building* includes shrubs too, as previously explained. *post* groups posts, tree trunks, and people. *vegetation* includes bicycles. Therefore, these classes mix elements of different nature, to some extent. There is, however, a consistent point underlying them, discussed previously in their presentation: the geometry. A definitive judgement on the semantic interpretation of the discovered classes depends on the target task. In the case where geometry constitutes the required information, these classes can be considered relevant. Otherwise, a finer classification system would be necessary, one that could for instance join shrubs to vegetation, and exclude bicycles from vegetation. Such a finer system would require as input more specific, detailed geometric representations, or other types of information such as vision or information from a knowledge-base.

Table 4.2 shows the quantitative results of the test. Figure 4.12 shows the classification on the test scans. The total F_1 obtained is 0.74. It is lower than the score obtained in the validation step, 0.77, yet close enough to confirm that the system was able to generalize from the validation set to the test set. A generalization at the dataset level, in this case, is therefore verified.

These observations confirm that the system is saturated in terms of its complexity. It is clearly not over-fitted. Instead, the performance obtained is the best that can be achieved under the current complexity. In order to increase the performance, the complexity of the system must be increased, by either using a more complex feature extraction stage

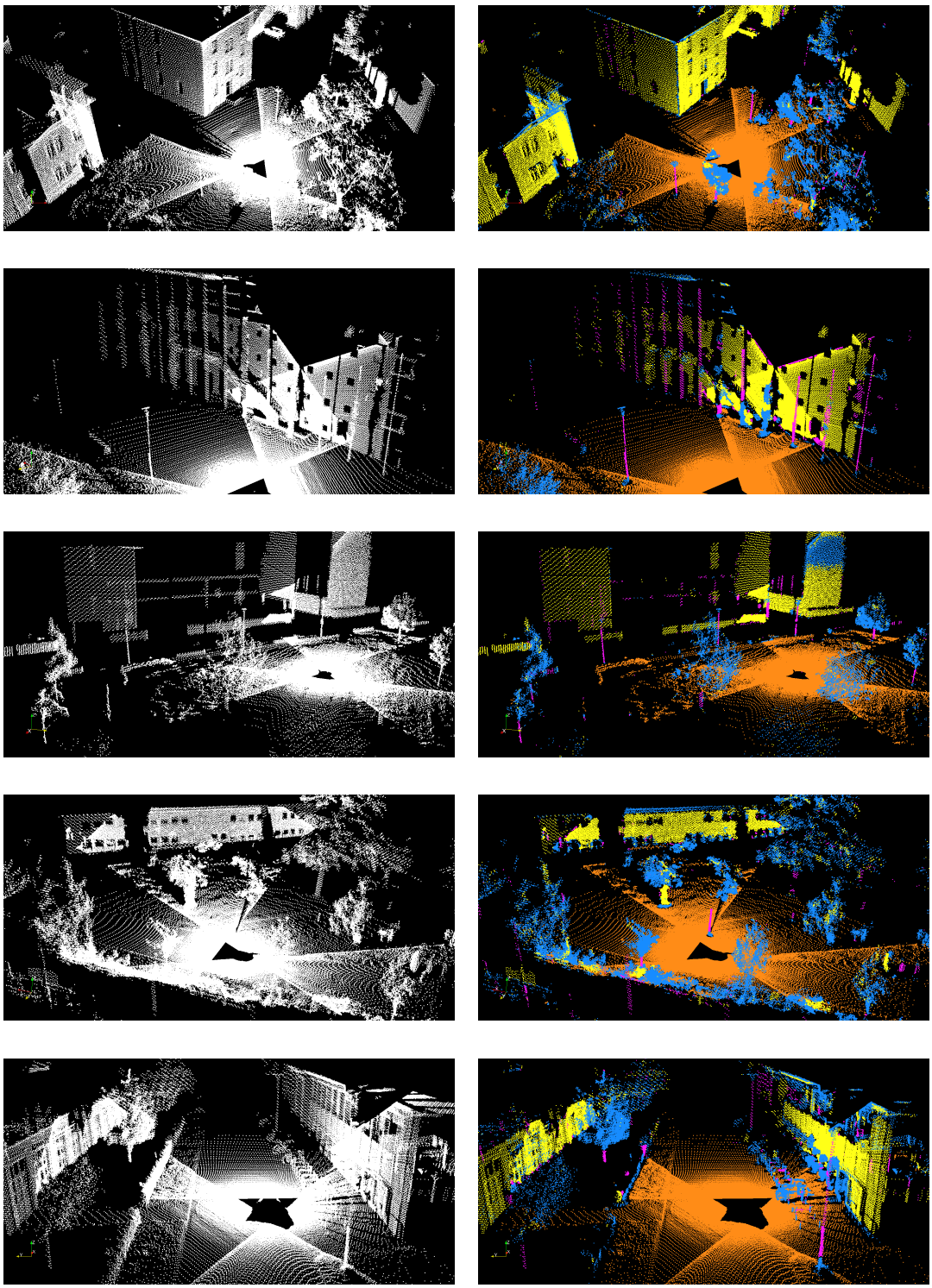


Figure 4.12: *Freiburg Classification Test Results.* Colours: (*ground*, orange), (*building*, yellow), (*post*, pink), (*vegetation*, blue). The misclassification of some facade regions as *vegetation* can be seen in some of the scans, as well as the misclassification of the base of posts as *vegetation*.

	unk	*ground*	*building*	*post*	*vegetation*	*rec*	$\boldsymbol{F_1}$
ground	4424	387200	110	54	1933	0.98	**0.98**
building	34955	248	93438	3961	22120	0.60	**0.73**
post	4212	55	1733	7614	4809	0.41	**0.50**
vegetation	27389	4986	7405	666	106657	0.73	**0.75**
pre	-	0.99	0.91	0.62	0.79	-	$\boldsymbol{F_{1_{total}} = 0.74}$

Table 4.2: *Freiburg Test Results.* Accuracy = 0.83.

or a more complex classification stage. Regarding the feature extraction, the selected system already presents the feature parameters that obtained the best performance, so the feature extraction process itself must be improved. As for the number of intermediary classes, since it is already at its maximum, complexifying the classification model means changing the model itself.

As for the remaining of the test scores, we can note that the precision scores are all higher than the recall scores. The main reason behind this difference is the impossibility of using all the intermediary classes provided by the GMM, leaving some of them as *unknown*. This can be observed in the results: apart from the case of the class *post*, the highest number of false negatives always appears under *unknown*. This is a characteristic of our two-layer approach: the data-oriented, intermediary layer learns the different patterns encountered in the data, but the task-oriented, final layer discards those which fail to correspond to some meaningful semantics.

Classwise, *ground* obtained the best F_1, precision and recall scores. This is understandable, because in the structured Freiburg environment, the ground can be consistently distinguished due to its planar shape and upwards normal orientation. *post* obtained the worst scores. It is confused with borders of other objects, especially corners of buildings, windows and doors. Additionally, the base and the top of tree trunks, as well as the base of posts, are frequently misclassified as *vegetation*. This factor can be spot in the confusion matrix, appearing as the high number of *vegetation* false positives actually corresponding to *post*. In all these cases, the confusions between *post* and *building* or *vegetation* have a greater negative effect on *post*, because of its rarity. *vegetation* suffers from misclassifications too, being confused with elements such as corners and roofs of buildings.

The limitations encountered are, in a way, due to the important density changes present in the data. Such changes make the distinctions harder by multiplying the patterns corresponding to each element. However, this point is addressed in part through the set of intermediary classes, which are able to represent some of the different patterns. Overall, the main problem seems to lie in the variability of the complex facade features, such as

windows, doors, roofs, prominent regions, corners and so on. These are the elements most frequently misclassified, either as post or vegetation, irrespective of sampling densities. The source of this problem, in turn, are the features employed, which do not allow a better distinction of the elements.

4.7.2 Caylus Results

From the validation results, we retrieve that the setups which obtained the best performances were, denoting the multi-scale system simply by R:

- regarding the training sets, {10-scan-set, $r = 0.6m$, $N_{CY} = 50$}, {5-scan-set, R, $N_{CY} = 50$} and {10-scan-set, R, $N_{CY} = 50$};
- regarding the feature radii, {5-scan-set, $r = 0.6m$, $N_{CY} = 50$} and {5-scan-set, R, $N_{CY} = 50$};
- regarding the number of intermediary classes, {5-scan-set, $r = 0.6m$, $N_{CY} = 30$}, {5-scan-set, $r = 0.6m$, $N_{CY} = 50$} and {5-scan-set, R, $N_{CY} = 50$}.

This leaves us with the single-scale setups {10-scan-set, $r = 0.6m$, $N_{CY} = 50$}, {5-scan-set, $r = 0.6m$, $N_{CY} = 50$}, and {5-scan-set, $r = 0.6m$, $N_{CY} = 30$}. Among these, the two last ones have equal performances, but compare unfavorably to the first one, according to the training set analysis. Therefore the best single-scale system is {10-scan-set, $r = 0.6m$, $N_{CY} = 50$}. Concerning the multi-scale setups, we are left with {5-scan-set, R, $N_{CY} = 50$} and {10-scan-set, R, $N_{CY} = 50$}. Because smaller training sets are preferable, the best system is {5-scan-set, R, $N_{CY} = 50$}. The two final systems, single and multi-scale, have not been compared before. This is done as follows: the single-scale has lower recall on *road* and *vegetation*, whereas the multi-scale has lower recall on *building*, *trunk* and *grass*. The single-scale system {10-scan-set, $r = 0.6m$, $N_{CY} = 50$} achieved thus the best performance.

The set of final classes, {*road*, *building*, *trunk*, *vegetation*, *grass*, *rough*}, is larger than the Freiburg set. *road*, *building* and *grass* have the purest semantic interpretations. *trunk* includes the road, posts and people. *vegetation* includes high grass, bushes and foliage. *rough* includes rough terrain, such as stony ground, and medium grass. As in the Freiburg case, the common underlying link is the geometry. It is interesting to note that the classes *ground*, *grass*, *rough* and *vegetation* can be interpreted as a progression in terms of scatterness of the geometry, while, to some extent, still correspond to specific elements in the environment.

Similarly to the Freiburg case, the test performance matched the validation performance, so the system was able to generalize to the test set, confirming its capacity to

achieve a dataset-level generalization. As for Freiburg, the results correspond to the system complexity. Because the best parameters were already determined through the validation, increasing the performance would require a change in the feature extraction stage or in the classification stage.

Figure 4.13 shows the classification results on the scans of the test set. The main problem is the non-distinction of the points nearest to the sensor, which normally correspond either to the road or to grass. In other words, the system was unable to separate nearby road from nearby grass. Another problem was that, at far range, vegetation was perceived as grass. Yet another difficulty encountered was one also present in the Freiburg dataset: the confusion between vegetation, trunks and building features. In the Caylus case, however, this effect was more constrained. Firstly, because the Caylus facades are sampled in a relatively similar manner, while the Freiburg facades are sampled in a variety of ways due to the overlapping scans. Secondly, because the Caylus facade features are simpler, corresponding to simple squared windows and doors.

The nonuniform sampling in the data impacted the distinction between distant road and distant buildings. At far range, because of the line-based sampling pattern of the Velodyne lidar, these two elements appear simply as lines, therefore having the same shape and normals oriented along unpredictable directions. The sampling also played a part in the confusion between distant vegetation and grass. Were the sampling denser, these two elements would maybe have been better distinguished. Clutter, on the other side, affected the classification of trunks, as many were considered as vegetation because they were entirely surrounded by it.

In the end, among the class confusions, the nonuniform sampling effects, and the clutter, the main source of difficulty remains the class confusions: nearby road with grass, distant vegetation with grass, buildings with trunks and vegetation. These, in turn, are a consequence of the feature representation used. Thus, as happened in the Freiburg case, the features do not allow a better distinction between these elements.

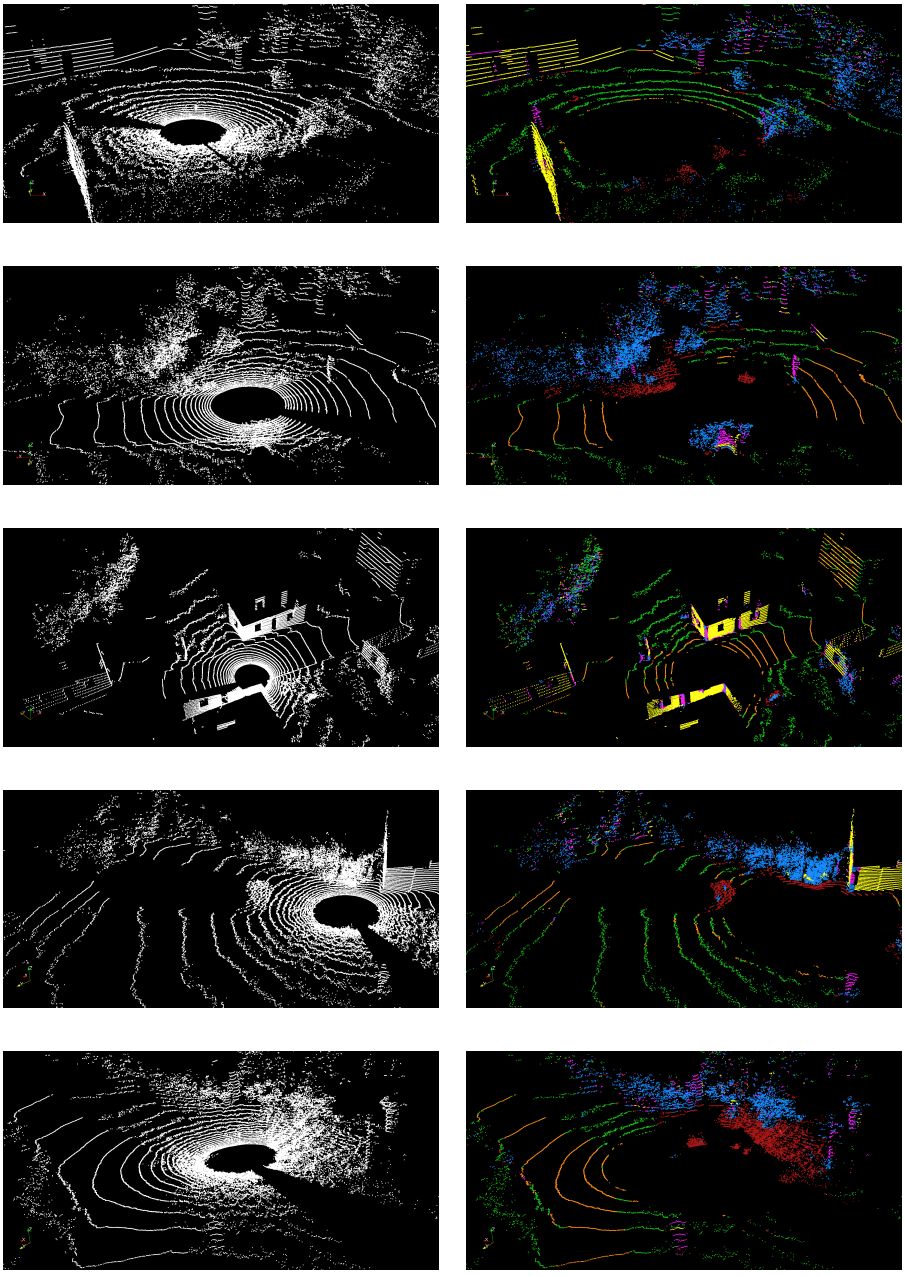


Figure 4.13: *Caylus Classification Test Results.* Colours: (*road*, orange), (*building*, yellow), (*trunk*, pink), (*vegetation*, blue), (*grass*, green), (*rough*, brown). The missing nearby points can be clearly detected in the scans. Some misclassifications of building features as *trunk* can also be observed.

Chapter 5

Conclusion

The approach proposed in this thesis consists of a classification system with a feature extraction stage and a two-layer classification model. It takes as input a 3D point cloud with its corresponding sensor-to-world transformation, and outputs a classified version of the point cloud. The feature extraction represents the shape of a point neighbourhood using information from a PCA operation. Regarding the classifier, the first, intermediary layer corresponds to a GMM trained in an unsupervised manner, and the second, final layer corresponds to a grouping of the intermediary classes into final classes. The approach avoids the necessity of manual labelling the input dataset, instead requiring a manual training of the grouping which is based on the trained intermediary layer.

The evaluations show that the approach is able to achieve a dataset generalization. The main advantages are the following:

+ Data-orientation. The unsupervised training of the GMM brings the data-oriented aspect to the system. The GMM classes are able to capture, at least in part, the variety of patterns in the data, addressing challenges such as non-uniform sampling, environment variability and clutter in an unsupervised manner.

+ Task-orientation. In spite of its unsupervised core, the approach is able to deliver final classes which, under certain conditions, can be semantically interpreted. In all cases, the final classes are consistent with the geometry represented through the features. The final classes are defined with the grouping, which brings the task-oriented aspect, in a principled and explicit manner. The system can be used as a predictive model in a target task.

+ Standard design and operation. The design follows the standard learning procedure of training, validation and test. The operation follows the standard classification pipeline. It is composed by a feature extraction and a classification stages. The classification stage groups the elementary stages of inference and decision.

+ Simplicity. There is no requirement of pre-processing or post-processing stages, although these may be included. The feature extraction and classification stages are simple. The design procedure, heavily based on unsupervised learning, is simple.

The main disadvantages are the following:

- Training noise. This refers to two factors. The first is the randomness present in the GMM training due to the random initialization method, constituting the GMM training noise. The second is the randomness present in the grouping training due to the expert supervision process, which is bound to be erroneous from time to time, constituting the grouping noise. Together, they both constitute the training noise. Such noise is a disadvantage because it affects the final classification performance, which ideally would be deterministic and predictable in all cases.

- Supervision computational complexity. The approach requires a supervised training of the grouping, which is done for each system evaluated in the learning process. In terms of computational complexity, this corresponds to a linear complexity. A supervised approach, on the contrary, requires a single, supervised labelling of the input dataset, offering a lower computational complexity. If taken individually, on the other hand, the grouping training is much simpler than the dataset labelling, because it is guided by the intermediary classes discovered by the GMM.

- Final performance. Considering the simplicity of the system, the performance is reasonable. With respect to the state-of-the-art, however, it does not compete with more advanced classification systems.

Between the feature extraction, the intermediary GMM and the final grouping, the feature extraction constitutes the main factor limiting the performance of the approach. In this context, a first extension could be the addition of an extra feature value, x_4, coming also from the PCA operation: the z coordinate of the world-frame representation of v_2, the second PCA eigenvector. This value can be seen as the natural complement for the current value x_3, which is the z coordinate of the world-frame representation of the normal, or v_3. This would add one extra dimension of information about the orientation of the points. It would allow, for instance, to isolate vertical linear elements such as vertical tree trunks and posts, with x_3 and x_4 being both zero in this case.

It would be interesting to test the approach under a full application case. A natural case would be terrain traversability analysis. A rough scheme of how our system can be used in such context is given hereafter. The final classes are each linked to a traversability class, which in turn are each linked to a traversability cost. The output pointwise structure is transformed to a 3D-voxel model. Under a conservative criterion, the class of a voxel

may be set as the highest-cost class among the classes of the individual points within the voxel. For a 2D path planning, the voxels above a certain height may be ignored, and the others projected onto a subsequent 2D-grid model, using the same conservative criterion. For a 3D path planning, the 3D-voxel model would already be enough. The classes of the final model, corresponding to traversability costs, would provide the cost criterion for a path planning algorithm. As an example, we regard the model produced for the Caylus dataset as being appropriate to tackle such problem in that environment, although at a global level, that is, merging many individual scans into a global map.

Besides terrain traversability analysis, the proposed approach may also be useful in a number of other cases. An example is the rapid production of an operational semantic model in a case of search and rescue. The simplicity of design of the system would be advantageous. The model produced could be used as a preliminary but operational model, while maybe a finer, more complete model would be put under construction. Another example is the comparison of different classification systems. In this case, our system could be used as a baseline, reference in the comparison, its simple and unsupervised nature allowing it to be rapidly designed. Yet another example is the labelling of a dataset, or equivalently, the production of a ground-truth, which, after all, was one of the main concerns behind this work. Because our system is unsupervised at its core, it can justly be used as an aid in the full labelling of a dataset. Thus, even if our approach does not lead to the final classification system, it may assist in its construction. Lastly, our approach can, of course, be used in any case where it already provides a fine-enough model for the required target task.

The previous observations about possible applications of our approach also lead to an important, but sometimes undervalued point: the best, most complete evaluation of any semantic modelling system is its testing under the target task for which it was designed. In other words, "the definitive quality of a product is measured by the client's satisfaction with it". This seems evident, but it is often overlooked in classification works, including ours. The reason for this importance is that the generic evaluation criteria used in classification, such as precision-recall-F_1, are not able to predict the system's performance at the actual application. Considering the case of terrain traversability analysis, for example, it is clear that using just the precision-recall-F_1 metrics is not enough to predict wether the robot will be able to correctly assess traversability and plan its path in all cases. It would be necessary to look for finer details, because a false positive could be enough to lead to disastrous results, while a false negative could be enough to restrain the robot of finding a path. Counting the percentage of times a robot was able to correctly find a path could serve as a more definitive evaluation of the system, compared to the standard classification metrics.

A last point is worth mentioning: the relation between supervision and lifelong learn-

ing. In learning, supervision is a very precious resource, because it provides the core information as how to perform a task, while at the same time being demanding in the point of view of the human domain expert. It is a constant goal, therefore, to optimize the amount of supervision. This becomes especially important in the light of lifelong learning, because it may be expected that a lifelong learning system will require episodic, unlimited supervised interventions to guide its development. To keep the process scalable, the system must thus be designed not only to accept these future interventions, but to ease them. This point was brought out during the implementation of our approach, when it became clear that the grouping training, despite being a simple supervised process, was still demanding when done multiple times.

Another form of minimizing supervision is to employ unsupervised learning. For a lifelong learning system, this seems to be extremely valuable, since it would allow it to develop and adapt to some modifications in the environment which do not necessariliy need any supervised guidance. The relevance and capacity of an unsupervised process were recalled in our work, which showed that the unsupervised GMM was indeed able to discover the pertinent patterns present in the features. However, unsupervised methods possess the inherent limitation of not producing directly interpretable results. This happens because they adapt to the data and to the prior only. A system learns by qualifying its output relatively to an external reference. The learning is relative to the reference. Unsupervised learning alone does not necessarily converge as desired, because it does not have the desired classes as its reference. It only has the data and the prior, which cannot be more than an implicit approximation of the reference.

It seems as if it is only by exploiting supervised and unsupervised learning together that lifelong learning could be approached. In this context, it would be mostly interesting to explore interactive techniques to do so, making systems which are truly designed to change, opposed to systems designed to converge towards a definitive model. Change would be brought in an interactive, episodic and flexible manner, by unsupervised processes as well as by optimized supervised processes.

Bibliography

Martin Adams, John Mullane, and Ba-Ngu Vo. Laser and radar based robotic perception. *Foundations and Trends in Robotics*, 1(3):135–252, 2011.

Ahmad Aijazi, Paul Checchin, and Laurent Trassoudaine. Segmentation based classification of 3d urban point clouds: A super-voxel based approach with evaluation. *Remote Sensing*, 5(4):1624–1650, 2013.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.

Jens Behley, Volker Steinhage, and Armin Cremers. Performance of histogram descriptors for the classification of 3d laser range data in urban environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4391–4398, 2012.

Jeff Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report TR-97-021, International Computer Science Institute, 1997.

Andreas Birk, Narunas Vaskevicius, Kaustubh Pathak, Sören Schwertfeger, Jann Poppinga, and Heiko Bülow. 3-d perception and modeling. *IEEE Robotics & Automation Magazine*, 16(4):53–60, 2009.

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

Johann Borenstein, H. Everett, Liqiang Feng, et al. Where am i? sensors and methods for mobile robot positioning. Technical Report 119.120.15, University of Michigan, 1996.

Nicolas Brodu and Dimitri Lague. 3d terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68(0):121–134, 2012.

Arun Das, Michael Diu, Neil Mathew, Christian Scharfenberger, James Servos, Andy Wong, John Zelek, David Clausi, and Steven Waslander. Mapping, planning, and sample

detection strategies for autonomous exploration. *Journal of Field Robotics*, 31(1):75–106, 2014.

Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

Karl Granström, Thomas Schön, Juan Nieto, and Fabio Ramos. Learning to close loops from range data. *The International Journal of Robotics Research*, 30(14):1728–1754, 2011.

Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2013.

Gideon Guy and Gérard Medioni. Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1265–1277, 1997.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2013.

Michael Himmelsbach, Thorsten Luettel, and Hans-Joachim Wuensche. Real-time object classification in 3d point clouds using point feature histograms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 994–1000, 2009.

Armin Hornung, Kai Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

Bradford King. *Range Data Analysis by Free-space Modeling and Tensor Voting*. PhD thesis, Rensselaer Polytechnic Institute, 2008.

Klaas Klasing, Daniel Althoff, Dirk Wollherr, and Martin Buss. Comparison of surface normal estimation methods for range sensing applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3206–3211, 2009.

Sven Koenig and Maxim Likhachev. D* lite. In *National Conference on Artificial Intelligence (AAAI)*, pages 476–483, 2002.

Jean-Franois Lalonde, Nicolas Vandapel, Daniel 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.

Ee Lim and David Suter. 3d terrestrial lidar classifications with super-voxels and multi-scale conditional random fields. *Computer-Aided Design*, 41(10):701–710, 2009.

Paul McManamon. Review of ladar: a historic, yet emerging, sensor technology with rich phenomenology. *Optical Engineering*, 51(6):060901–1, 2012.

Frank Moosmann and Miro Sauerland. Unsupervised discovery of object classes in 3d outdoor scenarios. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1038–1044, 2011.

Frank Moosmann, Oliver Pink, and Christoph Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *IEEE Intelligent Vehicles Symposium*, pages 215–220, 2009.

Kenneth Moreland. The paraview tutorial. Technical report, Sandia National Laboratories, 2013.

Daniel Munoz, Nicolas Vandapel, and Martial Hebert. Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2009–2016, 2009.

Frank Neuhaus, Denis Dillenberger, Johannes Pellenz, and Dietrich Paulus. Terrain drivability analysis in 3d laser range data for autonomous robot navigation in unstructured environments. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2009.

Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

Panagiotis Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373–1385, 2013.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Andrzej Pronobis. *Semantic Mapping with Mobile Robots*. PhD thesis, KTH Royal Institute of Technology, 2011.

Alastair Quadros, James Underwood, and Bertrand Douillard. An occlusion-aware feature for range images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4428–4435, 2012.

Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix, and Michel Devy. Rt-slam: A generic and real-time visual slam implementation. In *Computer Vision Systems*, volume 6962 of *Lecture Notes in Computer Science*, pages 31–40. Springer, 2011.

Michael Ruhnke, Bastian Steder, Giorgio Grisetti, and Wolfram Burgard. Unsupervised learning of compact 3d models based on the detection of recurrent structures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2137–2142, 2010.

Radu Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science Department, Technische Universitaet Muenchen, Germany, 2009.

Radu Rusu and Steve Cousins. 3d is here: Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, 2011.

Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: Science and Systems (RSS)*, 2009.

SICK. Lms200-30106. Datasheet, 2015a.

SICK. Lms291-s05. Datasheet, 2015b.

Bastian Steder, Michael Ruhnke, Slawomir Grzonka, and Wolfram Burgard. 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, 2011.

Elena Stumm, Andreas Breitenmoser, Franois Pomerleau, Cedric Pradalier, and Roland Siegwart. Tensor-voting-based navigation for robotic inspection of 3d surfaces using lidar point clouds. *The International Journal of Robotics Research*, 31(12):1465–1488, 2012.

Sebastian Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millennium*, pages 1–35. Morgan Kaufmann San Mateo, CA, 2003.

Rudolph Triebel, Rohan Paul, Daniela Rus, and Paul Newman. Parsing outdoor scenes from streamed 3d laser data using online clustering and incremental belief updates. In *AAAI Conference on Artificial Intelligence*, pages 2088–2095, 2012.

Ranjith Unnikrishnan. *Statistical Approaches to Multi-Scale Point Cloud Processing*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2008.

Velodyne. Velodyne's hdl-64e: A high definition lidar sensor for 3-d applications. White Paper, 2007.

Velodyne. Hdl-32e. User's Manual, Revision E, 2012.