

TESIS DOCTORAL del Programa de Doctorado en Ingeniería Mecatrónica

# **Contributions to Intelligent Scene Understanding of Unstructured Environments from 3D Lidar Sensors**

Victoria Plaza Leiva



UNIVERSIDAD DE MÁLAGA

Directores: Dr. Jose Antonio Gómez Ruiz

Dr. Antonio Mandow Andaluz


Tutor: Dr. Alfonso J. García Cerezo





UNIVERSIDAD  
DE MÁLAGA

AUTOR: Victoria Plaza Leiva

 <http://orcid.org/0000-0002-5944-762X>

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:

<http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>

Cualquier parte de esta obra se puede reproducir sin autorización pero con el reconocimiento y atribución de los autores.

No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): [riuma.uma.es](http://riuma.uma.es)



A mis padres Loli y Emilio  
A Alberto



UNIVERSIDAD  
DE MÁLAGA



## Agradecimientos - Acknowledgements

Porque el tiempo es lo más valioso que tenemos, el mérito de esta tesis pertenece sobretudo a todas las personas que han ayudado al trabajo como tal y/o que me han apoyado personalmente durante el largo camino, muchas veces invirtiendo el tiempo que no tenían.

En primer lugar, a aquellos que me apoyaron en la dura decisión de venir a Málaga arriesgando la comodidad de mi posición personal y laboral en Almería: mis padres, mi hermano, mi abuelo Salva por su sabiduría ante duras decisiones, y sobre todo a Raúl por su apoyo incondicional independientemente de su inconveniencia. También reconocer la inspiración de autosuperación y altruismo para seguir estudiando y aprendiendo *siempre* porque "el saber no ocupa lugar" heredada de mi padre y mi abuelo Diego.

A mis amigos Ángel y Jesús que, pese a la distancia, me ayudaron con mis dudas personales y laborales con sus consejos, los que me gustaba oír y los que no, siempre con acierto. Mis compañeras Mentxu y Ana han estado para escucharme y sacarme una sonrisa rompiendo mis "bucles". Y por supuesto gracias a Alberto por compartir su luz y su tiempo conmigo.

En lo laboral recuerdo con cariño las horas que robé a Jose Antonio Polaino, Javier Serón, Vicente Arévalo y a Fackhr-Eddine Ababsa que nunca tuvieron problemas en compartir sus conocimientos. Gracias a Anibal Ollero y Bernardine Dias por facilitarme la estancia en Estados Unidos. A mis compañeros de la pecera del laboratorio, mucha gente ha pasado por aquí en estos años y todos me aportaron algo bueno.

Gracias también, al apoyo técnico del Departamento de Ingeniería de Sistemas y Automática, profesores y personal de servicios, incluso los conserjes y personal de Seguridad del la Escuela de Ingenierías me han dado ánimos y han padecido con humor mis despistes, peticiones, y deshoras en el trabajo. Y por supuesto este trabajo no habría sido posible sin la financiación del Ministerio de Economía y Competitividad de España para mi beca predoctoral (BES-2012-061907), el proyecto nacional asociado (DPI2011-22443) y mis estancias de investigación en el extranjero: en el Robotics Institute de la Universidad Carnegie Mellon y en la Universidad de Evry Val d'Essone.

A José Antonio Gómez Ruíz por ser el guía principal de este trabajo confiando en mí, a Anthony Mandow por ayudarnos especialmente con la redacción y formalización de publicaciones, y a Jesús Morales por su colaboración en este trabajo.

Por último, agradecer a mi tutor, Alfonso García Cerezo, por haber confiado en mí y darme la oportunidad de realizar esta tesis con él, por facilitarme los recursos materiales y contactos personales a su disposición y además por tratarme como familia, apoyándome en los malos momentos y celebrando conmigo los buenos.

Más que un agradecimiento quiero dar la enhorabuena a todos ellos por el fruto que hemos obtenido juntos como equipo.

## Abstract

Three-dimensional (3D) lidar sensors are a key technology for navigation, localization, mapping and scene understanding in unmanned vehicles and mobile robots. This technology, which provides dense point clouds, can be especially suitable for novel applications in unstructured or natural environments, such as search and rescue, planetary exploration, agriculture, and off-road exploration. This is a challenging research area that involves disciplines ranging from sensor design to artificial intelligence and machine learning. In this context, this thesis proposes contributions towards intelligent scene understanding of unstructured environments from 3D ground-based range measurements. Particularly, the major contributions include new methodologies for classification of spatial features, object segmentation and navigability assessment in natural and urban environments, as well as the design and development of a new rotating multi-beam lidar (MBL).

Classification of spatial features is very relevant because this is largely required as a fundamental step towards higher level scene understanding problems. The thesis contributions in this respect aim to improve the effectiveness, both in computational load and accuracy, of supervised learning classification of spatial shape features (i.e., tubular, planar or scatter shapes) obtained from principal component analysis (PCA). This is achieved by proposing an efficient voxel-based neighborhood concept in an original contribution that defines offline training and online classification procedures as well as five alternative PCA-based feature vector definitions. Moreover, the feasibility of this approach is evaluated by implementing four types of supervised learning classifiers found in scene processing methods: a neural network model, support vector machines, Gaussian processes, and Gaussian mixture models.

Object segmentation is a further step towards scene understanding where sets of 3D points corresponding to the ground and other scene objects are isolated. The thesis proposes novel contributions for point cloud segmentation based on geometrically featured voxel maps. In particular, the proposed methodology involves two steps: first, a ground segmentation method specially designed for natural environments; second, subsequent isolation of individual objects. Besides, the ground segmentation method is integrated into a novel navigability map approach based on occupancy grids which can be suitable for mobile robots in natural environments.

The design and development of a new affordable high-resolution 3D lidar sensor is also proposed in the thesis. New commercial MBLs, such as those developed by Velodyne, are becoming an affordable and popular type of 3D sensor that offers high data rates within a limited vertical field of view (FOV). The proposed design consists on a tilting platform which improves the resolution and vertical FOV of the 16-beam Velodyne VLP-16. Furthermore, the complex scanning patterns produced by rotating MBL configurations are analyzed both in hollow sphere simulations and actual scans in representative environments.

## Resumen

Los sensores lidar 3D son una tecnología clave para navegación, localización, mapeo y entendimiento de escenas en vehiculos no tripulados y robots moviles. Esta tecnología, que provee nubes de puntos densas, puede ser especialmente adecuada para nuevas aplicaciones en entornos naturales o desestructurados, tales como búsqueda y rescate, exploración planetaria, agricultura, o exploración fuera de carretera. Esto es un desafío como área de investigación que incluye disciplinas que van desde el diseño de sensor a la inteligencia artificial o el aprendizaje automático (*machine learning*). En este contexto, esta tesis propone contribuciones al entendimiento inteligente de escenas en entornos desestructurados basado en medidas 3D de distancia a nivel del suelo. En concreto, las contribuciones principales incluyen nuevas metodologías para la clasificación de características espaciales, segmentación de objetos, y evaluación de navegabilidad en entornos naturales y urbanos, y también el diseño y desarrollo de un nuevo lidar rotatorio multi-haz (MBL).

La clasificación de características espaciales es muy relevante porque es extensamente requerida como un paso fundamental previo a los problemas de entendimiento de alto nivel de una escena. Las contribuciones de la tesis en este respecto tratan de mejorar la eficacia, tanto en carga computacional como en precisión, de clasificación de aprendizaje supervisado de características de forma espacial (forma tubular, plana o difusa) obtenida mediante el análisis de componentes principales (PCA). Esto se ha conseguido proponiendo un concepto eficiente de vecindario basado en voxel en una contribución original que define los procedimientos de aprendizaje *offline* y clasificación *online* a la vez que cinco definiciones alternativas de vectores de características basados en PCA. Además, la viabilidad de este enfoque es evaluado mediante la implementación de cuatro tipos de clasificadores de aprendizaje supervisado encontrados en metodos de procesamiento de escenas: red neuronal, máquina de vectores de soporte, procesos gaussianos, y modelos de mezcla gaussiana.

La segmentación de objetos es un paso más allá hacia el entendimiento de escena, donde conjuntos de puntos 3D correspondientes al suelo y otros objetos de la escena son aislados. La tesis propone nuevas contribuciones a la segmentación de nubes de puntos basados en mapas de voxeles caracterizados geoméricamente. En concreto, la metodología propuesta se compone de dos pasos: primero, una segmentación del suelo especialmente

diseñado para entornos naturales; y segundo, el posterior aislamiento de objetos individuales. Además, el método de segmentación del suelo es integrado en una nueva técnica de mapa de navegabilidad basado en cuadrícula de ocupación el cuál puede ser apropiado para robots móviles en entornos naturales.

El diseño y desarrollo de un nuevo y asequible sensor lidar 3D de alta resolución también se ha propuesto en la tesis. Los nuevos MBLs, tales como los desarrollados por Velodyne, están siendo cada vez más un tipo de sensor 3D asequible y popular que ofrece alto ratio de datos en un campo de visión vertical (FOV) limitado. El diseño propuesto consiste en una plataforma giratoria que mejora la resolución y el FOV vertical de un Velodyne VLP-16 de 16 haces. Además, los complejos patrones de escaneo producidos por configuraciones de MBL que rotan se analizan tanto en simulaciones de esfera hueca como en escáneres reales en entornos representativos.

# Table of contents

<b>List of figures</b>	<b>xv</b>
<b>List of tables</b>	<b>xvii</b>
<b>Nomenclature</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background and motivation . . . . .	1
1.2 Contributions . . . . .	3
1.3 Context . . . . .	6
1.4 Thesis outline . . . . .	9
<b>2 Related work</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 3D classification methods . . . . .	12
2.2.1 Supervised learning methods for point cloud classification . . . . .	14
2.2.2 Geometric features extraction . . . . .	16
2.3 3D data processing for scene understanding . . . . .	17
2.3.1 3D representation . . . . .	17
2.3.2 Segmentation . . . . .	19
2.3.3 Navigability assessment . . . . .	20
2.4 Customized 3D lidar sensors . . . . .	22
<b>3 General framework for supervised learning classification of lidar point clouds based on voxel-based neighborhood</b>	<b>27</b>
3.1 Introduction . . . . .	27
3.2 General voxel-based neighborhood framework for geometric pattern classification . . . . .	28
3.2.1 Definitions . . . . .	28



3.2.2	Training and classification procedures . . . . .	28
3.2.3	Extracting spatial shape features from voxels . . . . .	30
3.2.4	Data structures . . . . .	31
3.3	Experimental setup and methodology . . . . .	32
3.3.1	Experimental datasets . . . . .	32
3.3.2	Classifiers parametrization . . . . .	33
3.3.3	Methodology . . . . .	36
3.4	Performance analysis and comparison . . . . .	37
3.4.1	Performance evaluation with linear combination of eigenvalues . . . . .	37
3.4.2	Performance evaluation with different feature vector definitions . . . . .	37
3.4.3	Computation time . . . . .	38
3.4.4	Comparison with point-wise neighborhood classification . . . . .	40
3.5	Summary . . . . .	42
<b>4</b>	<b>Segmentation and navigability assessment based on geometrically featured voxel maps</b>	<b>45</b>
4.1	Introduction . . . . .	45
4.2	Segmentation methods for unstructured environments . . . . .	47
4.2.1	Ground segmentation . . . . .	47
4.2.2	Object segmentation . . . . .	50
4.2.3	Evaluation . . . . .	51
4.3	Navigability assessment . . . . .	58
4.3.1	Evaluation . . . . .	60
4.4	Summary . . . . .	61
<b>5</b>	<b>Analysis and construction of a multi-beam lidar with an additional degree of freedom</b>	<b>65</b>
5.1	Introduction . . . . .	65
5.2	Rotation of a multi-beam lidar sensor . . . . .	66
5.2.1	Commercial multi-beam lidars . . . . .	66
5.2.2	Multi-beam lidar with an additional rotation . . . . .	67
5.3	Analysis methodology . . . . .	68
5.3.1	Numerical simulation . . . . .	69
5.3.2	Sampling density . . . . .	69
5.4	Analysis of ideal RMBL scanning patterns and comparison with other 3D configurations . . . . .	70
5.4.1	Qualitative analysis . . . . .	70



---

5.4.2	Sampling density . . . . .	72
5.5	Implementation of a portable tilting mechanism for a Velodyne VLP-16 . .	74
5.5.1	Velomotion-16 system description . . . . .	75
5.5.2	Analysis of the scan measurement distribution for Velomotion-16 .	77
5.6	Discussion of example scans . . . . .	79
5.7	Summary . . . . .	82
<b>6</b>	<b>Conclusion and future work</b>	<b>85</b>
6.1	Conclusion . . . . .	85
6.2	Future work . . . . .	88
	<b>Resumen de la tesis</b>	<b>89</b>
	<b>References</b>	<b>103</b>
	<b>Appendix A 3D data processing</b>	<b>113</b>
A.1	Robotic operating system . . . . .	113
A.2	Velodyne HDL-32E . . . . .	117
A.3	Unolaser . . . . .	125
A.4	Velomotion . . . . .	126



UNIVERSIDAD  
DE MÁLAGA

# List of figures

2.1	Mobile laser scanning with a 2D LRF . . . . .	11
2.2	Range image produced in stereo vision . . . . .	12
2.3	RGB-D scene produced by combining a 3D scanner and a CCD camera . . . . .	13
2.4	Geometric features based on PC . . . . .	16
2.5	Spherical subsampling . . . . .	18
2.6	Customized lidar system based on a 2D LRF with two new rotations . . . . .	22
2.7	Customized lidar system based on a 2D LRF with a complete rotation in roll . . . . .	24
3.1	Classification procedures with voxel-based neighborhood computation . . . . .	29
3.2	Hand labeled <i>Urban</i> point cloud. . . . .	33
3.3	Hand labeled <i>Natural_1</i> point cloud. . . . .	34
3.4	Hand labeled <i>Natural_2</i> point cloud. . . . .	34
3.5	Hand labeled <i>Garden</i> point cloud. . . . .	34
3.6	<i>Urban</i> point cloud classified by NN with $\mathcal{F}_2$ . . . . .	39
3.7	<i>Natural_1</i> point cloud classified by NN with $\mathcal{F}_2$ . . . . .	40
3.8	<i>Natural_2</i> point cloud classified by NN with $\mathcal{F}_2$ . . . . .	41
3.9	Geometrically featured voxel map from a natural environment. . . . .	43
4.1	General description of the segmentation and navigability assessment solution . . . . .	45
4.2	Flow diagram of the ground segmentation algorithm. . . . .	47
4.3	Illustration of ground segmentation rules . . . . .	49
4.4	Scene segmented into objects with no adjacent voxels between them. . . . .	51
4.5	Scene with objects segmented into regions with the same geometric class . . . . .	52
4.6	Errors in segmentation: oversegmentation and undersegmentation. . . . .	53
4.7	Dense scene segmented with the method <i>ABDC</i> . . . . .	55
4.8	Slope scene wrongly segmented. . . . .	56
4.9	Segmentation results with different voxel sizes . . . . .	57
4.10	Execution times with different voxel sizes . . . . .	58



4.11	Flow diagram of the occupancy grid generation . . . . .	59
4.12	Occupancy grid resulting in a trees scene . . . . .	61
4.13	Occupancy grid resulting in a slope scene . . . . .	62
5.1	Commercial multi-beam lidars . . . . .	66
5.2	Reference frames of Velodyne VLP-16 and Velomotion-16. . . . .	67
5.3	Representation of lidar beams as sphere points. . . . .	70
5.4	Different views of the sphere points given by the RMBL . . . . .	71
5.5	Orthogonal plane points of lidar sensors . . . . .	72
5.6	2D histogram of beam density on the sphere . . . . .	73
5.7	Beam density histogram . . . . .	74
5.8	Views of the Velomotion-16 RMBL sensor . . . . .	74
5.9	<i>Velomotion-16</i> reference frames and tilting parameters. . . . .	75
5.10	Velomotion-16 system architecture. . . . .	77
5.11	Analysis of 3D scan measurement distribution for <i>Velomotion-16</i> . . . . .	78
5.12	Photos of experimental scenes . . . . .	79
5.13	Scans from an indoor scene . . . . .	81
5.14	Scans from the urban scene . . . . .	81
5.15	Scans from an outdoor terrain scene . . . . .	82
A.1	Topic based communication in ROS . . . . .	114
A.2	Service based communication in ROS . . . . .	115
A.3	ROS code hierarchy . . . . .	116
A.4	Example of a PCD file . . . . .	117
A.5	Velodyne HDL-32E . . . . .	117
A.6	Velodyne capture shown in <i>DSR viewer</i> . . . . .	118
A.7	PCAP file from Velodyne shown by <i>Wireshark</i> . . . . .	119
A.8	Structure of an Ethernet distance packet from Velodyne . . . . .	120
A.9	Rviz ROS viewer showing a Velodyne point cloud . . . . .	124
A.10	Unolaser LRF . . . . .	125
A.11	Point cloud captured by Unolaser . . . . .	126
A.12	Velomotion communication with ROS. . . . .	127

# List of tables

2.1	Representative examples of customized lidar systems based on a commercial device with an extra DOF. . . . .	23
3.1	Characteristics of hand labeled voxels of experimental point clouds . . . . .	35
3.2	Performance of classifiers ( $MCC$ and $\bar{M}$ ) using feature vector definition $\mathcal{F}_2$ for the evaluation datasets. . . . .	37
3.3	Performance of classifiers ( $MCC$ ) for the training dataset with different definitions of the feature vector $\mathcal{F}$ . . . . .	38
3.4	Performance of classifiers ( $MCC$ and $\bar{M}$ ) for the evaluation datasets using selected feature vector definition ( $\mathcal{F}_2$ for NN, $\mathcal{F}_3$ for SVM and $\mathcal{F}_4$ for GMM and GP). . . . .	39
3.5	Computation times for training and classification, in seconds. . . . .	40
3.6	Performance of GMM classifier with point-wise neighborhood using feature vector definition $\mathcal{F}_2$ for the evaluation datasets. . . . .	41
3.7	Computation times for point-wise neighborhood training and classification, in seconds. . . . .	41
4.1	Segmentation Results with Different Ground Segmentation Rule Sequences	53
5.1	Manufacturer specifications for the VLP-16 and HDL-32 sensors . . . . .	66
5.2	Specifications of Velomotion-16 . . . . .	76
5.3	Sensor performance in example scans. . . . .	80
A.1	PCD header structure . . . . .	116
A.2	PCAP file structure from Velodyne . . . . .	120





UNIVERSIDAD  
DE MÁLAGA

# Nomenclature

## Acronyms / Abbreviations

<i>MCC</i>	Matthew's Correlation Coefficient
2D	Bi-dimensional
3D	Three-dimensional
DOF	Degree of freedom
EC	Electronically communicated
EM	Expectation Maximization
FH	Felzenszwalb and Huttenlocher segmentation algorithm
FOV	Field of view
GFVM	Geometrically featured voxel map
GMM	Gaussian mixture model
GP	Gaussian processes
KNN	<i>k</i> -Nearest neighbors
LRF	Laser rangefinder
MBL	Multi-beam lidar
MLP	Multi-layer perceptron
Ncut	Normalized Cut segmentation algorithm
NN	Neural network

PCA Principal component analysis

PCL Point Cloud Library

RGB Red-green-blue color model representation

RGB-D Combination of a RGB image and its corresponding depth image

RMBL Rotating multi-beam lidar

ROS Robot Operating System

RSBL rotating single-beam lidar

SAR Search and rescue robot

SLAM Simultaneous localization and mapping

SMBL Spinning multi-beam lidar

SVM Support vector machine

TLS Terrestrial lidar scanning



**Symbols**

$(x_i, y_i, z_i)$  Cartesian coordinates of  $X_i$

$[R, \omega, \alpha]$  Spherical coordinates: radial distance, azimuthal angle, and polar angle

$\eta_h, \eta_l$  High and low thresholds of the vertical collision zone for a vehicle

$\gamma$  Rotation angle of the additional rotating mechanism in a RMBL

$\lambda_0, \lambda_1, \lambda_2$  Eigenvalues in a covariance matrix in descending order

$\mathbb{F}$  Metric for false positives in ground segmentation

$\mathbb{G}$  Metric for accuracy in ground segmentation

$\mathbb{N}$  Novel metric for accuracy in objects segmentation with bigger penalty in undersegmentation

$\mathbb{O}$  Metric for accuracy in objects segmentation

$\mathbf{p}_i$   $i^{th}$  input pattern in a dataset

$\mathcal{C}$  Set of target classes

$\mathcal{D}$  Input dataset

$\mathcal{F}$  Feature vector definition

$\bar{c}$  Normalization of  $c$  in  $[0,1]$  with respect to a 95% confidence interval

$\bar{M}$  Normalized confusion matrix

$\bar{X}$  Average of 3D points in a set

$\vec{e}_0, \vec{e}_1, \vec{e}_2$  Eigenvectors corresponding to  $\lambda_0, \lambda_1, \lambda_2$  respectively

$\rho$  Threshold of minimum number of points in a voxel

$\theta_1, \theta_2$  Thresholds for geometric classification

$C_j$   $j^{th}$ -target class

$d$  Distance between RMBL origin and its optical center

---

$E$	Edge size of voxels
$G$	Set of voxels that correspond to ground points
$G_d$	Set of voxels disconnected to the lowest voxel in $G$
$G_l$	Set of voxels connected to the lowest voxel in $G$
$h_h, h_l$	High and low thresholds of the vertical collision zone for a vehicle considering the height of the current $XY$
$M$	Confusion matrix
$M_{ij}$	$i$ $j^{\text{th}}$ entry in $M$ . Is the number of elements of true class $i$ that have been assigned to class $j$ by the classifier
$N$	Number of patterns in an input space, such as number of points in a dataset or neighborhood
$N_C$	Number of target classes
$n_g^j$	Number of Gaussian components in a model $j$ of GMM
$N_L$	Number of components of $\mathbf{p}_i$
$N_V$	Number of voxel in a spatial 3D grid including all points in $\mathcal{D}$
$N_{95c}$	Rounded integer number of the 95% significant voxels in the middle of the distribution of $c$
$P$	Range measurement
$p_i^j$	Component $j$ of $\mathbf{p}_i$
$s()$	Size of an object, that is the number of its inner 3D points
$t_i$	Target class for $\mathbf{p}_i$
$V$	Data structure of a GFVM
$X_i$	$i$ -th 3D point in a set
$X_v Y_v Z_v$	Local frame of the VLP-16
$XY$	Cell in a 2D horizontal grid

U',V',W Objects in te scene segmented with the proposed algorithm

U,V,W Hand-labeled objects



UNIVERSIDAD  
DE MÁLAGA

# Chapter 1

## Introduction

### 1.1 Background and motivation

Automatic recognition of the environment is a complex process which is used in robots with autonomous tasks such as navigation or manipulation. In autonomous vehicles the accurate identification of traversable terrain and obstacles is crucial to the safe navigation. Furthermore, new challenging applications of autonomous vehicles, such as Search and Rescue or planetary exploration, take place in natural or unstructured environments. On-board exteroceptive sensors such as cameras or 3D scanners can produce accurate and high resolution information about the surrounding scene, but this huge amount of raw data may be difficult to understand for a decision making process. For this reason, complex sensor information must be processed so that it can be useful for navigation control. The main purpose of this thesis is to offer contributions to the challenging scene understanding problem in unstructured environments based on 3D point clouds. The major contributions include new solutions for point cloud classification, object segmentation, navigability assessment, and the development and qualitative analysis of a tilting multi-beam lidar device.

Due to the big amount of data captured from a 3D scene, the large point cloud can be transformed into a effective representation to reduce the execution time when processing the features extraction or posterior applications such as object classification or the trajectory planification. There are three main groups of techniques: *i*) reducing the data in the point cloud, for instance keeping only the frontal closest area [86] or discarding redundant information [57], *ii*) transforming the information into a 3D model such as elevation maps [92][56] which reconstruct the terrain in two and half dimensions loosing relevant geometric information from the scene, or voxel maps [17] which avoid that problem, or *iii*) extracting a bi-dimensional (2D) model such as height map [29] or occupancy grid [103] [23]. This thesis project proposes a voxel-based neighborhood computation where points in each non-

overlapping voxel in a regular grid are processed at once to extract relevant 3D features. This voxel-based solution significantly reduces data processing in comparison with point-based techniques.

Classification of primitive geometric features is largely used as a fundamental step towards higher level scene understanding problems [116]. For instance, classifying points into coarse geometric categories such as vertical or horizontal has been proposed as the first layer of a hierarchical methodology to process complex urban scenes [32]. Furthermore, classification of scan points prior to segmentation is useful to process objects with unclear boundaries, such as ground, vegetation and tree crowns [8]. In this sense, spatial shape features can describe the shape of objects for later contextual classification [127]. Thus, classification of spatial shape features based on principal component analysis (PCA) is a constituent process in recent scene processing methods [125][18][38][49][55]. Therefore, this thesis focuses on improving the effectiveness, both in computational load and accuracy, of supervised learning classification of spatial shape features (i.e., tubular, planar or scatter shapes) obtained from covariance analysis [47].

After processing the raw data from the scene, such as the voxelization and geometric classification, a segmentation and navigability analysis may be applied. A segmentation method is required to isolate individual objects in a scene as a prior process to the classification or semantic recognition of them. Much effort has been made to solve the problem of segmenting in 2D images, and more recently the challenge has also been applied to 3D point clouds, as 3D sensors have become more widely used. In [26] several segmentation algorithms have been developed for both dense and sparse point clouds, which depends on the scan device, concluding that algorithms which first extract the ground offer more accurate results in the subsequent segmentation process. Some works start the process extracting the ground by fitting it in a flat plane [126] [125] or by using a height threshold [122]. Then the non-ground points are grouped in objects or segments depending on the distance between points by using a graph based algorithm such as Normalized Cut or Ncut [101] [122] and the algorithm of Felzenszwalb and Huttenlocher (FH) [31] over the range image [126] or over the 3D point cloud [102]. Other methods perform the segmentation based on the similarity of the normal vector computed at each point taking into account the local neighbor points. In [89] the normal at each point is computed with the information of the  $k$ -Nearest Neighbors (KNN). To reduce the normal estimation at each point, in [125] the points are first clustered if they are nearer than a distance threshold, which is called the growing region technique, then these clusters are grouped based on the similarity of the normal. The main drawback of this point-based segmentation approaches is the high memory and execution time consumption since it has to process each point in the point cloud, hence they may be incompatible with

Real-Time applications. Another set of techniques for 3D segmentation is the group-based techniques, which aim to solve the time-consumption problem of the point-based techniques by reducing the scene to a voxel model. Usually the scene is discretized in 3D cubes with constant size as a 3D grid, but these voxels can also be dynamically positioned around groups of points [37]. Once the ground has been extracted, the objects are isolated by merging adjacent voxels [25]. Other authors do not extract the ground as a previous step, but implement segmentation based on geometric information as local geometric features (eigenvalues from the covariance matrix, or PCA) [97] or besides that, also based on height and normal vector [38] based on voxels or point-wise neighborhood. The main problem of these techniques is that they assume flat and continuous ground, as in an urban environment. Thus ground detection is risky in unstructured environments where big slopes or non-flat areas belong to the ground.

The commercial devices for obtaining dense and accurate 3D data are expensive, heavy and/or power consuming. Most of the solutions to minimize the cost of a 3D scanner device consist in rotate a 2D laser rangefinder (LRF) in any axis [59] [63]. Less authors [50] and [72] use a rotation with an already finished 3D LRF (Velodyne) for mapping applications. The Velodyne Lidar sensors are being increasingly used in many 3D laser scanner applications. However, the point clouds generated by multi-beam scanners, as Velodyne, have a sparse and inhomogeneous density. This mentioned point cloud inhomogeneity substantially affects the results of typical processing techniques, such as finding good point correspondences for scan registration [111]. Due to the density rises around the new rotation axis, [121] implements not only one as usual but two additional rotations, in pitch and roll, to make uniform the density. This thesis proposes adding a tilting rotation to the Velodyne Puck with the purpose of increasing density, which is made more homogeneous, as well as the vertical field of view. In particular, a compact and light design is presented which can be used as a portable sensor. This design has been implemented as the new *Velomotion-16* 3D sensor.

This thesis addresses challenges of the robotic scene understanding problem regarding point cloud classification, segmentation of natural terrain and other scene objects, and navigability assessment. Besides, the thesis explores the possibilities offered by the addition of a degree of freedom to multi-beam lidar sensors to obtain affordable rapid full-3D high resolution scans.

## 1.2 Contributions

The goal of this thesis is to contribute new methodologies and developments towards intelligent scene understanding of unstructured environments based on 3D lidar sensor information.

This sensor technology, which provides dense point clouds, can be especially suitable for novel applications in unstructured or natural environments, such as search and rescue and off-road exploration. This is a challenging research area that involves disciplines ranging from sensor design to artificial intelligence and machine learning. Contributions include new methodologies for classification of spatial features, object segmentation and navigability assessment in natural and urban environments, as well as the design and development of a new rotating multi-beam lidar (MBL).

In particular, this PhD dissertation offers the following main contributions:

- **A voxel-based neighborhood approach for classification of spatial features with supervised learning methods.** Improving the effectiveness of spatial shape features classification from 3D lidar data is very relevant because it is largely used as a fundamental step towards higher level scene understanding challenges of autonomous vehicles and terrestrial robots. In this sense, computing neighborhood for points in dense scans becomes a costly process for both training and classification. The thesis contributes an effective voxel-based neighborhood computation where points in each non-overlapping voxel in a regular grid are assigned to the same class by considering features within a support region defined by the voxel itself. Thus, the classified point cloud can be represented by the resulting geometrically featured voxel map. Besides, the thesis proposes a new general framework for implementing and comparing different supervised learning classifiers. These contributions provide offline training and online classification procedures as well as five alternative feature vector definitions based on principal component analysis for scatter, tubular and planar shapes. Moreover, the feasibility of this approach is evaluated by implementing a neural network (NN) method, in particular a multi-layer perceptron-based model, as well as three other supervised learning classifiers found in scene processing methods: support vector machines (SVM), Gaussian processes (GP), and Gaussian mixture models (GMM). This voxel-based solution significantly reduces data processing in comparison with point-based techniques, hence it may be compatible with real-time applications.
- **Segmentation and navigability analysis based on geometrically featured voxel maps (GFVM).** Navigability assessment is useful for planning obstacles free trajectories in autonomous navigation tasks in 3D environments. Based on GFVM, the proposed methodology involves two steps: first, a ground segmentation method specially designed for natural environments; second, subsequent isolation of individual objects. Besides, the ground segmentation method is integrated into a novel navigability map approach based on occupancy grids which can be suitable for mobile robots



in natural environments. The unstructured ground is extracted, that is, each voxel is identified as ground or non-ground considering slopes and rough surfaces found in unstructured scenes. Then, the scene is split into objects and a second segmentation in regions inside each object is performed based on the voxel's geometric class. Finally, the proposed 2D occupancy grid represents accessible areas by considering the attributes and the height of each voxel over the ground level in relation to vehicle dimensions.

- **Design and development of a portable and affordable 3D lidar sensor.** The Velodyne multi-beam lidars are being increasingly used in many 3D laser scanner applications. However, the point clouds generated by multi-beam scanners have a sparse and inhomogeneous density, which is sparser for the less expensive and lighter VLP-16 (Puck) sensor. This limitation is partially coped with in mobile applications by considering longitudinal motion in successive scans, but remains a problem in stop and go or static applications. This thesis proposes adding a tilting rotation to the Velodyne Puck with the purpose of increasing density, which is made more homogeneous, as well as the vertical field of view. Even if adding an extra degree of freedom has been common with two-dimensional scanners, this idea has not been exploited with light 3D multi-beam devices. In particular, a compact and light design is presented which can be used as a portable sensor. This design has been implemented as the new *Velomotion-16* 3D sensor.

The scientific production of this thesis is listed below, starting from the most recent ones:

- [66] Sensors. Analysis of 3D Scan Measurement Distribution with Application to a Multi-Beam Lidar on a Rotating Platform. 2018. (**Journal Q1**).
- [88] Sensors. Voxel-Based Neighborhood for Spatial Shape Pattern Classification of Lidar Point Clouds with Supervised Learning. 2017. (**Journal Q1**).
- [15] HRI' 16, IEEE International Conference on Human-Robot Interaction. NavCue: Context immersive navigation assistance for blind travelers. 2016.
- [87] MED' 15, 23rd Mediterranean Conference on Control and Automation. Occupancy grids generation based on Geometric-Featured Voxel maps. 2015.
- [85] IWANN' 15, 13th International Work-Conference on Artificial Neural Networks. Multi-layer Perceptrons for Voxel-based Classification of Point Clouds from Natural Environments. 2015. (Core B)

- [84] ICIT' 15, IEEE International Conference on Industrial Technology. 3D Segmentation Method for Natural Environments based on a Geometric-Featured Voxel Map. 2015.
- [86] Master Thesis for Master in Mechatronics, University of Málaga. Local Semantic Map based on 3D data for navigability. 2014.
- [83] Eurathlon' 14, Workshop and Summer School euRathlon Arcas. Local Semantic Map based on 3D data for navigability. 2014.
- [98] ROBOT' 13, First Iberian Robot Conference. CUADRIGA: Robot Móvil para Búsqueda de Víctimas en Situaciones de Emergencia (Quadriga: A Mobile Robot for Ground Search and Rescue). 2013.

### 1.3 Context

This thesis has been conducted within the Robotics and Intelligent Control Systems research line of the *Doctoral Program in Mechatronics Engineering* (Programa de Doctorado en Ingeniería Mecatrónica) of Universidad de Málaga (UMA), established as an official Doctoral Program by the Spanish Government (RUCT registration number 5600225).

This doctoral thesis has been supported by a doctoral grant (BES-2012-061907) given by the Spanish Ministry of Economy and Competitiveness (MINECO) associated with the National Project RAMBLER: Towards Long-Range Exploration Robot Autonomy in Natural Environments (DPI2011-22443). This project aims to develop and implement a set of new tools to advance towards the goal of a long-range exploration robot. These tools include the generation of maps of the environment with different levels of interpretation, such as cartography, occupation, slopes, admissible trajectories classification, as well as the inclusion of semantic information for the interpretation of elements different to the ground. It will also address new and robust 3D simultaneous localization and mapping (SLAM) mechanisms to allow long time operation under limited supervision in natural environments.

Furthermore, two research stays were also supported by MINECO in the context of research stay grants for doctoral students. The author spent a three-month stay at the group IRA2 (Interaction, Réalité Augmentée, Robotique Ambiante), laboratory of IBISC (Informatique, Biologie Intégrative et Systèmes Complexes) of the University of Évry Val d'Essonne in France continuing with her thesis project activity under the supervision of Phd. Mr. Fakh-Eddine Ababsa. The second research stay, with a duration of four months, was at the Robotics Institute of Carnegie Mellon University in Pittsburgh (Pennsylvania, United

States of America) under the supervision of Phd. Mrs. M. Bernardine Dias and Phd. Mr. Aaron Steinfeld. During that stay the author participated in the project *Assistive Robot for Blind Travelers*, which is described below in the list of all research projects related to this thesis. In the course of both stays, the author produced as outcomes the publications [84] and [15], respectively, mentioned in Section 1.2.

The author has developed her major research within the Robotics and Mechatronics Lab of the Systems Engineering and Automation Department (Departamento de Ingeniería de Sistemas y Automática) of UMA. The research interests of this group include the implementation of robotics systems for emergency response in catastrophe situations with the goal of improving the response time and safety of the rescue team. Some main achievements of the group were the design and implementation of mobile robots that successfully assist in human rescue (Alacrane: [65], Quadriga: [98]). In addition, the group focused its research on the development of low cost sensors to capture the 3D environment on real time (Unolaser [63] and Unomotion [59]) and the accurate recognition of the terrain for autonomous navigation [56]. Thus, this PhD dissertation is a step forward in natural terrain recognition which could be applied to robotic autonomous navigation such as search and rescue missions.

Besides, the author has participated in other research projects related to the thesis:

- *ATICA: All Terrain Intelligent Compact and Autonomous Vehicle* (806/56.3879). Funded by CDTI and Grupo ITURRI S.A. from 02/07/2012 to 31/12/2014. Development of an off-road autonomous vehicle for transport applications in outdoor environments. The vehicle contains onboard sensors (laser, GPS, IMU) and the control system is based on several processors with Ubuntu and the well-known Robot Operating System (ROS) platform. My contribution is the integration of the 3D laser rangefinder information with the navigation system through the generation of an occupancy grid with labelled obstacles depending on its height.
- *Autonomous navigation of a 4x4 mobile robot in natural terrains based on differential GPS and 3D laser rangefinder* (P10-TEP-6101-R). Funded by 'Proyecto de Excelencia de la Junta de Andalucía' from 27/03/2013 to 27/03/2017. My contribution was the development of an algorithm for the Quadriga robot integrated with the motion control system in Labview [98]. The algorithm aims to locate victims by: *i*) an autonomous navigation algorithm of complete coverage in a determined area, integrated with Google Maps, and *ii*) tracking the victims telephone Bluetooth signals.
- *First-Rob: Multi-Robot system for cooperation with first response human and canine rescue teams in catastrophe scenarios* (DPI2015-65186-R). Funded by MINECO from 01/01/2016 to 31/12/2018. The mission of the project is to develop a comprehensive

and user-centered mechatronic multi-robot system to cooperate with First Response teams with the ultimate goals of reducing the risks for workers and improving their efficiency by collecting and assessing relevant information from the disaster zone under difficult perceptual conditions prior to the decision of sending rescue workers. This project extends the Rambler project associated to my doctoral thesis. My contribution is the development of a novel MBL.

- *Assistive Robot for Blind Travelers* (National Robotics Initiative, IIS-1317989). Funded by National Science Foundation at Carnegie Mellon University. During my research stay in CMU I developed a facial recognition algorithm using ROS and OpenCV (cascade classifier) for the Baxter humanoid robot in order to customize the interaction of Baxter with each different user. The main contribution is the change of communication channel (voice, screen or gestures) of the multimodal user depending on the sensitive disability of the user (blind, deaf, mute) or his/her preferences. The program involves the creation and update of a data base with the users (face and preferred communication channel) in real time during the interaction. In addition, I also developed a customized greeting with the name of the user and the handshake (user hand location recognized by vision). Furthermore, I collaborated in the sub-project of NavCue, an intelligent module for providing rich, multi-sensory, context-based information for travelers who are blind or low vision. The robot gestures the route (A\* path-planning algorithm) accompanied by spoken directions including sensory landmarks and cues which provides redundancy and memorable features for more confident navigation.

For the training period of my thesis I have attended the following courses:

- Master on Mechatronics in the University of Málaga (January 2013- July 2014).
- Summer School on Mechatronics, University of Brno (September - October 2013).
- Advanced CUDA training course for Nvidia GPUs, University of Málaga (December 2013 - January 2014).
- Mechatronic Systems - Introductory Course, University of Málaga (June 2014).
- Improving The Efficiency of SFM-SLAM by using depth information, University of Málaga (June 2014).
- euRathlon/ARCAS Workshop and Summer School on Field Robotics (June 2014).
- Workshop Data Structures for Large-Scale 3D Point Cloud Processing. The 13th International Conference on Intelligent Autonomous Systems, IAS13 (July 2014).

## 1.4 Thesis outline

This thesis is divided into six chapters, one appendix, and bibliographical references. Except for this chapter and the one related to the conclusions and future work, each chapter starts with an introduction that states the addressed problem and ends with a summary that highlights the contributions and/or the results that have been obtained.

Chapter 2, *Related work*, offers an up-to-date state of the robotic solutions applied to the topics or tasks involved on this thesis. There is a revision of the most common 3D representation techniques, classification methods, segmentation, occupancy grids, and lidar sensors.

Chapter 3, *General framework for supervised learning classification of Lidar point clouds based on voxel-based neighborhood*, describes the proposed solution using different classifiers and parametrization, and analyze the results and performance obtained in experiments.

Chapter 4, *Segmentation and navigability analysis based on geometrically featured voxel maps*, describes some applications for the classification method for 3D point clouds proposed in Chapter 3. Segmentation and occupancy grid algorithms are described and tested in natural scenarios.

Chapter 5, *Analysis and construction of a multi-beam lidar with an additional degree of freedom*, describes a new developed low cost 3D LRF and analyzes the complex scan patterns produced by this sensor configuration.

Chapter 6, *Conclusion and future work*, highlights the most relevant contributions of this thesis and proposes future research topics.

Finally, the appendix provides a further analysis of the software architecture of the software integrated in the ROS platform.



UNIVERSIDAD  
DE MÁLAGA

# Chapter 2

## Related work

### 2.1 Introduction

3D perception of the environment is crucial for autonomous vehicles. Some authors have addressed this issue with stereo vision [52] [5] [103] [120] (see Fig. 2.2) or a 2D LRF which creates 3D point clouds with the movement of the vehicle [18] (see Fig. 2.1) as affordable approaches. Recent 2D scanners have a multiecho capability which is very useful for detecting sparse objects as vegetation and still detect the solid object behind that [12]. However, the most common solution is a 3D Lidar [68] although its high cost. Some outdoor robots are equipped with 3D LRFs in combination with other sensors such as differential GPS, inertial units [21], both of them [61], or cameras [94] [25] [73] [41] [40]. Widely used is the RGB-D information which is the combination of the point cloud (or depth info) from a 3D lidar and an RGB image from a camera [119], as a result a colored point cloud is obtained (see Fig. 2.3).

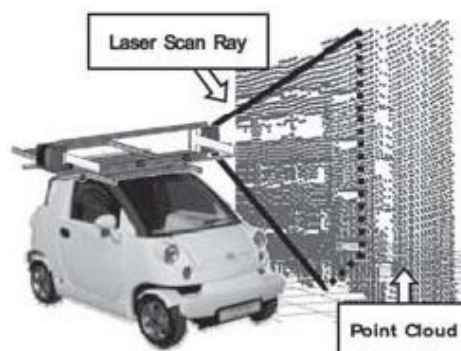


Fig. 2.1 Mobile laser scanning with a 2D LRF which accumulates consecutive scans while the vehicle is moving to produce a whole 3D point cloud [18].



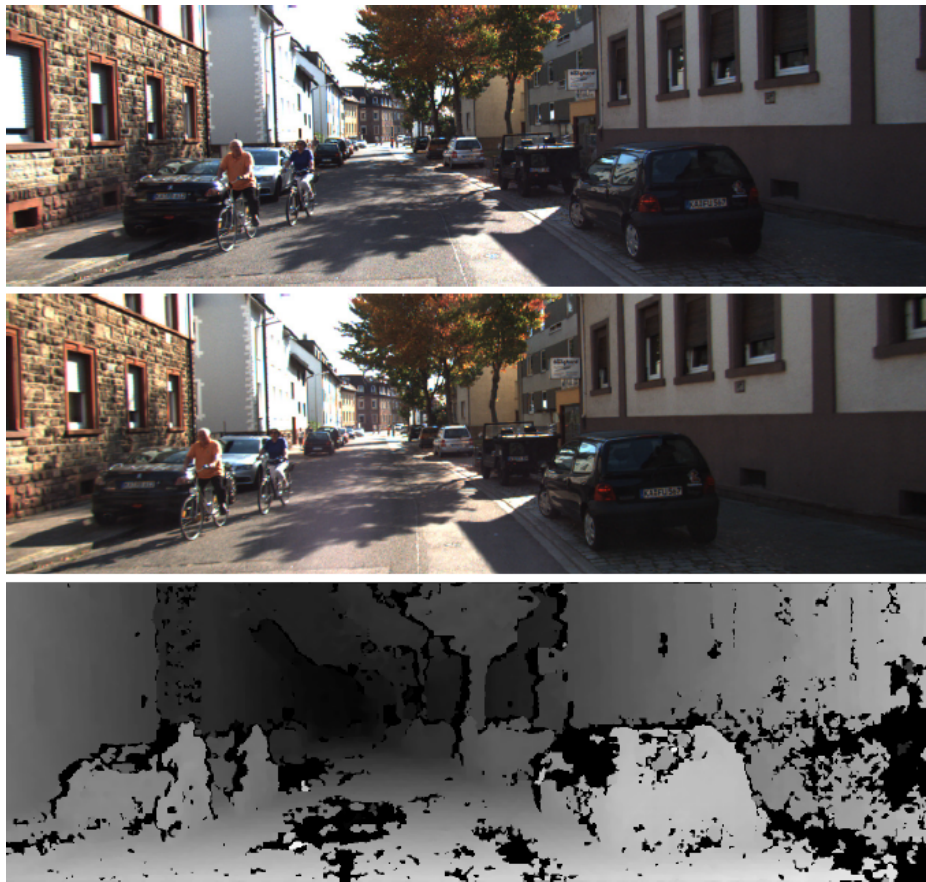


Fig. 2.2 Range image (or depth image) produced in stereo vision as the disparity from images in two cameras at the same time [120]

This chapter reviews the state-of-the-art of the major aspects of 3D lidar sensing addressed in the thesis. Section 2.2 analyzes different classifiers and describes the PCA method for extracting geometric features. Section 2.3 reviews different aspects of 3D point cloud processing, that include useful representation of 3D information, segmentation, and navigability assessment for autonomous vehicles. Finally, 2.4 discusses different commercial and customized 3D lidar systems.

## 2.2 3D classification methods

3D lidar sensors are a key technology for navigation, localization, mapping and scene understanding in novel ground vehicle systems such as autonomous cars [54], search and rescue robots [60], and planetary exploration rovers [99]. One major limitation regarding the use of lidar technology in these challenging applications is the time and computational resources required to process the dense point clouds generated by these sensors.





Fig. 2.3 RGB-D scene produced by combining information from a 3D scanner and a CCD camera [119]

Classification techniques involving point clouds are used extensively and can be categorized in many ways [49]. For instance, airborne sensors can use elevation and flatness characteristics to classify roof surfaces and urban objects [46][124][104] whereas terrestrial scans are affected by obstructions and varying point density [8]. Furthermore, algorithms have been proposed to identify particular object types, such as vehicles, buildings or trees [116][123], or to classify geometric primitives at point level [47]. In this sense, while some methods segment the cloud before classifying points within the resulting clusters [35][18], others perform classification directly on scan points [8]. Moreover, different machine learning descriptors have been considered (e.g., histograms [8][49] and conditional random fields [51][127]). In particular, many solutions rely on supervised learning classifiers such as SVM [35][126][73][38], GP [16] [95], or GMM [47][41][77][55].

Many classification techniques are point-wise in that they compute features for every point in a cloud by using the points within its local neighborhood, the support region. The KNN algorithm can produce irregular support regions whose volume depends on the varying sampling density of objects and surfaces from terrestrial scans [8]. For example, KNN has been used to compare the performance between several classifiers [35] and to classify into planar or non-planar surfaces [38]. The KNN support volume can be limited by setting a fix-bound radius [39]. Furthermore, ellipsoidal support regions of adaptive sizes, denoted as super-voxels, can be built iteratively based on point characteristics [51][79]. Other point-wise classification techniques adopt regular support regions by searching for all neighbors within a given radius [47][116][8][95]. In general, point-wise techniques imply a high computational

load. This is why some authors have proposed oversampling techniques to reduce the amount of data in the raw point cloud [51][48].

Grid representations and voxels have also been considered to speed up point cloud classification. In some solutions, grids serve to segment points prior to point-wise classification. For instance, the method proposed in [39] computes segmentation by projecting non ground points on a 2D grid and [37] uses voxels for defining groups of points that are later classified with a Neural Network (NN) supervised learning method. Some authors have proposed computing features for points in a voxel by considering support regions defined by neighboring voxels. The authors in [49] compute PCA for each voxel with a support region defined by the 26-neighbors. Descriptors are used both for segmentation (i.e., voxel clusters) and for later classification of the set of points within a cluster. Furthermore, in [18] the feature vector for each voxel is obtained from a support region that includes a number of surrounding voxels. In this case, features are not employed for classification but for mapping voxels to a color space used for segmentation. Neither [49] nor [18] compute features to classify points within a voxel.

### 2.2.1 Supervised learning methods for point cloud classification

This section briefly reviews supervised learning methods that have been used in the literature for point cloud scene classification: SVM, GP, GMM, and NN.

#### Support Vector Machine

The purpose of SVM learning [22] is to find a hyperplane that separates the dataset into a discrete predefined set of classes consistent with labeled training patterns. When patterns are not linearly separable, SVM transforms original data into a new space and uses a kernel function for classification. SVM has shown good generalization even with a reduced training dataset, although its performance can be significantly affected by parametrization [67]. Apart from the definition of the kernel function, SVM uses a box constraint, which is a parameter that controls the maximum penalty imposed on margin-violating observations and contributes to prevent overfitting.

SVM has been applied to classify urban point clouds into ground, and planar and non-planar points on the ground [38]. In this application, every point is evaluated together with its KNN based on covariance analysis that uses a linear combination of eigenvalues and a Radial Basis kernel function. Furthermore, the same kernel function with SVM has been applied to lidar data in intelligent vehicles to detect vegetation [73] and to classify clusters of points as urban objects [35][126][49].

### Gaussian Processes

GP is a generalization of the Gaussian probability distribution [90] that can be interpreted as a Bayesian version of the SVM method. Each class is modeled as a GP where a covariance function (kernel) is trained to estimate its nonparametric underlying distribution. The problem of learning in GP is exactly the problem of finding suitable parameters (called hyperparameters) for the covariance and mean functions that best model the training input dataset. Generally, the GP method requires defining the following: the number of function evaluations, a covariance function, an inference method, a mean function, a likelihood function, and the initialization of the hyperparameters.

GP has been applied for real-time ground segmentation by considering the relative height of lidar points from a land vehicle [16]. Moreover, a combination of GP and SVM with PCA has been proposed to classify terrain as traversable or non traversable by computing two features representing texture and slope for every point [95].

### Gaussian Mixture Model

A GMM is a probabilistic model that uses a mixture of Gaussian probability distributions to represent subpopulations within a population [28]. In the case of more than two classes, a different GMM is inferred for each class. Then, the learning algorithm tunes the weight, mean, and covariance matrices of a mixture of  $n_g$  Gaussian components for each GMM. The training process finds  $n_g$  for each GMM given a maximum value  $N_G$ .

Lalonde *et al.* [47] used GMM with the expectation maximization (EM) algorithm [11] to classify lidar points in scatter, planar and tubular classes according to saliency features. GMM has also been applied with color and spatial features for pixel-wise segmentation of road images [41] and object and background classification in point clouds [77].

### Neural Networks

The multi-layer perceptron (MLP) is a type of NN commonly used in supervised learning [80]. Implementing an MLP requires a definition of the network topology (i.e., the number of layers and neurons), the transfer function in every layer, the back-propagation learning algorithm, and the learning constant. .

MLPs has been used to classify urban objects from non-ground points distributed within point clusters [125] and voxels [37]. Furthermore, the problem of classifying vehicles represented as point clouds has been addressed with a combination of NN and genetic algorithms [117].

## 2.2.2 Geometric features extraction

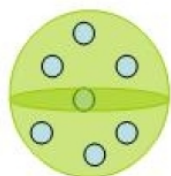
The classification of spatial shape relies on geometric features to classify each neighborhood as: planar surface, tubular structure, or scatter regions [47]. The local spatial distribution of the points within a neighborhood is obtained by decomposition in the principal components from the covariance matrix of the points' Cartesian coordinates. The positive symmetric covariance matrix for each neighborhood is determined by a set of the  $N$  inner 3D points  $X_i = (x_i, y_i, z_i)^T$  with average  $\bar{X} = \frac{1}{N} \sum_{i=1}^N X_i$ , and it is defined in the expression 2.1:

$$\frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})(X_i - \bar{X})^T \quad (2.1)$$

The matrix is decomposed into its main components and its eigenvalues which are sorted in ascending order,  $\lambda_0 \geq \lambda_1 \geq \lambda_2$ . Eigenvectors  $\vec{e}_0, \vec{e}_1, \vec{e}_2$  correspond to eigenvalues  $\lambda_0, \lambda_1, \lambda_2$  respectively. For the scatter voxels, it is satisfied that  $\lambda_0 \approx \lambda_1 \gg \lambda_2$  and no dominant direction can be found. For the tubular structure case, the main direction, tangent at the curve, is defined by the eigenvector  $\vec{e}_0$  associated to the dominant eigenvalue, that is  $\lambda_0 \gg \lambda_1 \approx \lambda_2$ . Finally, for the planar surfaces, the main direction is aligned with the normal surface, two eigenvalues are greater than the third one, that is  $\lambda_0 \approx \lambda_1 \gg \lambda_2$ , and  $\vec{e}_0, \vec{e}_1$  define the plane. A linear combination of the eigenvalues is used, see expression 2.2, to represent the three saliency features named *scatter-ness*, *tubular-ness* and *surface-ness*. Fig. 2.4 shows, the three output features, linear, surface and scatter, that are linear combinations of the eigenvalues.

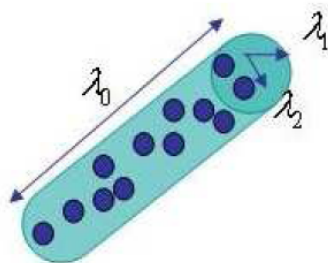
$$\begin{bmatrix} \text{scatter-ness} \\ \text{tubular-ness} \\ \text{plane-ness} \end{bmatrix} = \begin{bmatrix} \lambda_0 \\ \lambda_0 - \lambda_1 \\ \lambda_1 - \lambda_2 \end{bmatrix} \quad (2.2)$$

$$\lambda_0 \approx \lambda_1 \approx \lambda_2$$



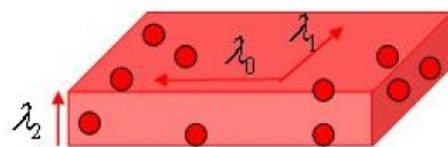
$$S_{\text{scatter}} = \lambda_0$$

$$\lambda_0 \gg \lambda_1 \approx \lambda_2$$



$$\vec{S}_{\text{linear}} = (\lambda_0 - \lambda_1) \vec{e}_0$$

$$\lambda_0 \approx \lambda_1 \gg \lambda_2$$



$$\vec{S}_{\text{surface}} = (\lambda_1 - \lambda_2) \vec{e}_2$$

Fig. 2.4 Geometric features based on PCA [47]

In practice, the decision of which class represents a voxel may be performed manually with thresholds. If  $\lambda_0 - \lambda_1$  is higher than a threshold  $\theta_1$  indicates *tubular - ness*, and  $\lambda_1 - \lambda_2$  higher than a threshold  $\theta_2$  represents *surface - ness*; otherwise *scatter - ness*. Nevertheless, [86] demonstrates that the manual adjustment of  $\theta_1$  and  $\theta_2$  thresholds for the classification process is not viable or desired because these values can vary depending on the scene, sensor characteristics, or point density.

An approach to make this process automatically is training a classifier to maximize the classification successful probability based on training data. In [47], a classifier based on a GMM was trained using the expectation maximization algorithm [11]. However, this classifier involves a higher computational cost than real time applications requires.

## 2.3 3D data processing for scene understanding

This section reviews published works that address different aspects of 3D point cloud processing: useful representation of 3D information, segmentation, and navigability assessment for autonomous vehicles.

### 2.3.1 3D representation

Due to the big amount of data captured from a 3D environment, the large point cloud can be transformed into an effective representation to reduce the execution time when processing object classification or the navigation map for the trajectory plan. There are three main groups of techniques: *i*) keeping the point cloud as the principal data structure but reducing the amount of data, *ii*) transforming the info into a 3D model, or *iii*) reducing the info into a 2D model.

Some navigation systems requires only the frontal closest area of the vehicle to decide the immediate next movement, hence information from the sides, back, and too far or too high areas (according to the robot height) can be discarded from the scene [86]. Other approach is keeping the 360° scene information but reducing redundant information, for instance by applying a spherical subsampling algorithm which produces a point cloud equally dense in all the directions [57] (see Fig. 2.5). It is a low computational cost approach for data reduction in spherical scanners which takes into account the scan direction resolutions.

The second group is to generate 3D model such as terrain model. Elevation maps model the natural terrain by using meshes, such as Delaunay triangulation [34] or irregular triangular meshes [92]. Nevertheless, these meshes models cannot reconstruct the surfaces in areas with no input data, and they are also non scalable algorithms. Markov random fields [81] and

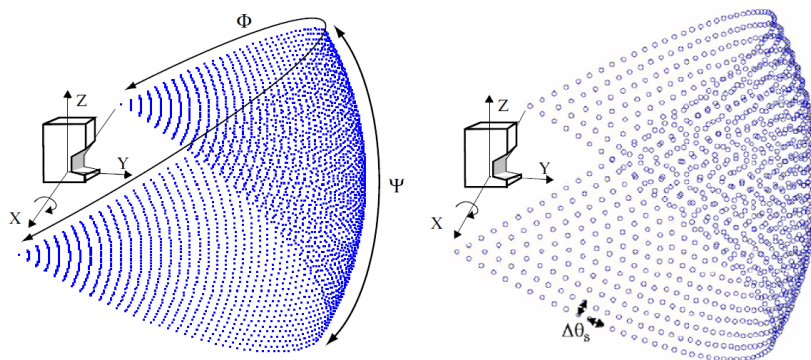


Fig. 2.5 Measure-direction density for (a) a raw scan, and (b) spherical point subsampling [57].

fuzzy logic [56] can reconstruct the surface with incomplete data. The main disadvantage of the terrain model is that it is not a complete 3D reconstruction of the scene but 2.5D. This is because it takes into account the highest point in the vertical, so overhangs [91] as the crowns of the trees are marked as an elevation over the terrain. Not only the terrain but the whole 3D scene can be modeled using geometric features. A VM [17] represents the scene with a resolution that is coarser than the actual size of the objects, which means that accessible regions may become non-accessible. The normal distribution [18] can be used but the execution time is too high for a real time processing in a rescue mission. These geometric feature techniques obtain better results in urban than in natural environments.

Other approaches for extracting semantic information from the scene do not use neither the 3D point cloud a 3D model but a simpler 2D data structure. A 3D scene can be also represented by a normal map and/or a height map [29]. A normal map is a color 2D image where each pixel has the representative color of the normal at this point. A height map is a grey scale 2D image where each pixel has the intensity proportional to the height, it means the distance between the capturing device and the object. In [19] an accurate edge detection has been implemented based on three different data structures: a normal map, the point cloud organized as a 2D matrix, and the RGB image captured by a camera at the same time. Furthermore, 3D texture analysis from some objects can be done by extracting a deviation map [75]. Firstly, a smooth surface is adjusted to the object geometric structure, then a deviation map can be obtained by computing the difference between the two surfaces. Therefore, the texture can be analyzed through a representative 2D image. The main advantage of these methods are that they can use well-known and optimized artificial vision algorithms for 2D images.



### 2.3.2 Segmentation

A segmentation method is required to isolate individual objects in a scene as a prior process to the classification or semantic recognition of them. Much effort has been made to solve the problem of segmenting in 2D images, and more recently the challenge has also been applied to the three-dimensional (3D) point clouds, as 3D sensors have become more widely used.

There are two main approaches for 3D segmentation. The first approach, called point-based techniques, is based on analyzing each point with features from its local neighborhood. Some works start the process extracting the ground by fitting it in a flat plane [126] [125] or by using a height threshold [122][53]. Then the non-ground points are grouped in objects or segments. These groups can be obtained depending on the distance between points by using a graph based algorithm such as Ncut [101] [122] and the algorithm of FH [31] over the range image [126] or over the 3D point cloud [102]. Other methods perform the segmentation based on the similarity of the normal vector computed at each point taking into account the local neighbor points. In [89] the normal at each point is computed with the information of the KNN. [4] combines a planar segmentation and a topological refinement based on graphs. To reduce the normal estimation at each point, in [125] the points are first clustered if they are nearer than a distance threshold, which is called the growing region technique, then these clusters are grouped based on the similarity of the normal. Alternatively, [38] the authors classify each point in ground, planar, and non-planar points through an SVM whose inputs are local geometric features at each point, with KNN. These features are: normal vector, height above the lowest point in the scene, or eigenvalues from the covariance matrix. The planar clusters are segmented in two stages: by using a Gaussian sphere, and then refining with a distance-based clustering method, that is merging if distances are similar. The non-planar points are clustered based only on distance. The main drawback of this segmentation approach is the high memory and execution time consumption since it has to process each point in the point cloud. Therefore, these methods may be incompatible with real-time applications.

Another set of techniques for 3D segmentation is the group-based techniques, which aim to solve the time-consumption problem of the point-based techniques by reducing the scene to a voxel model. Usually the scene is discretized in 3D cubes or voxels with constant size as a 3D grid, but these voxels can also be dynamically positioned around groups of points [37]. In [26] several algorithms have been developed for both dense and sparse point clouds, which depends on the scan device. The work concludes that algorithms which extract the ground offer more accurate results in the subsequent segmentation process and navigability assessment. In sparse point clouds, as acquired from a Velodyne scan, an interpolation method is performed to propagate the ground to areas without 3D information. On the

contrary, for dense point clouds, as obtained with a Riegl scan, the ground is detected as the largest area with adjacent voxels with height mean or a vertical variance less than a threshold. Once the ground has been extracted, the objects are isolated by merging adjacent voxels or voxels in a distance lower than a predefined threshold. In [25] the segmentation is also with adjacent voxels, and the results are contrasted and fused with an RGB data segmentation process. In addition, the ground is extracted through an elevation map. Other authors have approached ground segmentation by fitting a flat plane [126] [125] or by using a threshold, for the height mean [122] or for both the height mean and the vertical variance [26]. These approaches are useful for urban environments, but they do not consider the 3D complexity of natural terrain. The main problem of these techniques is that they assume flat and continuous ground, as in an urban environment. Hence, big slopes or non-flat areas are not grouped in the ground. A novel idea for the voxel map is introduced in [97] called geometric-featured voxel maps. In this model the voxels contain geometric information taken from a prior classification in linear or tubular structures, scattered shapes, horizontal planes, or vertical planes [17]. The ground is extracted as horizontal surface voxels with no other kind of voxels below. Horizontal surface voxels higher than a threshold over another identical kind of voxel in the same position are discarded. This way, slopes can be detected as the floor depending on the previous geometric classification, but some horizontal surfaces can be wrongly detected as ground. For instance, horizontal surface from an object may not have points below because of the shadows in the scene captured. Furthermore, another important disadvantage to this technique is that classification may not be enough since the surfaces are classified in vertical and horizontal, hence it requires a very accurate threshold for slopes. All these methods have been designed for urban environments and thus ground detection is risky in unstructured environments where big slopes or non-flat areas belong to the ground. [112] combines both approaches, elevation map and geometric model fitting methods, in an adaptive ground segmentation method with good results in uneven and sparse regions which frequently appear in natural scenes. However it is based in a data representation which is only affordable for 3D point cloud obtained by multibeam sensors, such as Velodyne.

### 2.3.3 Navigability assessment

Navigability assessment is performed as a previous step for subsequent task of motion planning. Autonomous mobile robots, such as search and rescue robots (SAR) or long-range explorer extraterrestrial robots, require an efficient trajectory avoiding obstacles and low computational cost in order to increase its velocity or to save energy, respectively or to invest processing resources in more complex and specific goals, such as the recognition of specific areas. Navigability assessment is addressed by three typical approaches: a



binary classification problem, traversable or non-traversable terrain, a finer estimation with a continuous traversability score, or a classification with various classes. [78] reviews this terrain traversability analysis problem and distinguishes the geometry-based approaches according to the criteria employed for estimating traversability. The extraction of 3D features which determine whether the areas from the scene are accessible cannot be solved in a unique step but with several complex algorithms combined. The information from the scene is usually obtained through laser scanners [57] and stereo vision [52] as large and complex point clouds. Processing all this data requires loads of time which may be incompatible with real-time applications such as SAR autonomous navigation. Therefore, the features extraction of the 3D scene and subsequently the occupancy grid generation require a simplified representation.

Three main approaches have been adopted for the extraction of 3D features. The first approach reduces the scene into a 2D representation, which can be processed with standard artificial vision algorithms, or into 2.5D elevation maps. Local and global statistics features of each object and ground can be obtained based on a range image [126] or a 2D deviation map [75]. Like 2D representations model only the frontal view of the scene, the elevation maps [56] model the terrain in a zenithal view, but not the whole scene, which is the main disadvantage of this approach.

The second approach consists on point-based techniques, where each 3D point is analyzed based on the features from its local neighborhood. In this case, the definition of the neighboring points for each point becomes a costly step in 3D object recognition. Every point is evaluated together with its  $k$ -nearest neighbors to extract geometric features and then is classified as planar or non-planar [38] or in tubular structures, planar surfaces, and scatter [47]. The main drawback of this approach is the memory and execution time required to process each point in the point cloud. Therefore, these methods may be incompatible with real-time applications such as SAR.

Another group of techniques for 3D features extraction is group-based techniques, which aim to solve the time-consumption problem of the point-based techniques by reducing the scene to a voxel model. Features are extracted at each neighborhood which means each voxel instead of the neighborhood at each point as the previous approach. Each voxel can be defined by its global height in the scene and the ground can be extracted through a height threshold [12]. Otherwise, voxels may be classified as tubular, scatter, horizontal surfaces or vertical surfaces, obtaining a geometric-featured voxel map where traversable ground can be detected [97]. However, these methods are restricted to urban environments where no slopes on the ground are assumed.

Many autonomous mobile robots generate a discrete model of occupancy or traversability in order to reduce the scene data for the path planning processing. Most of techniques divide

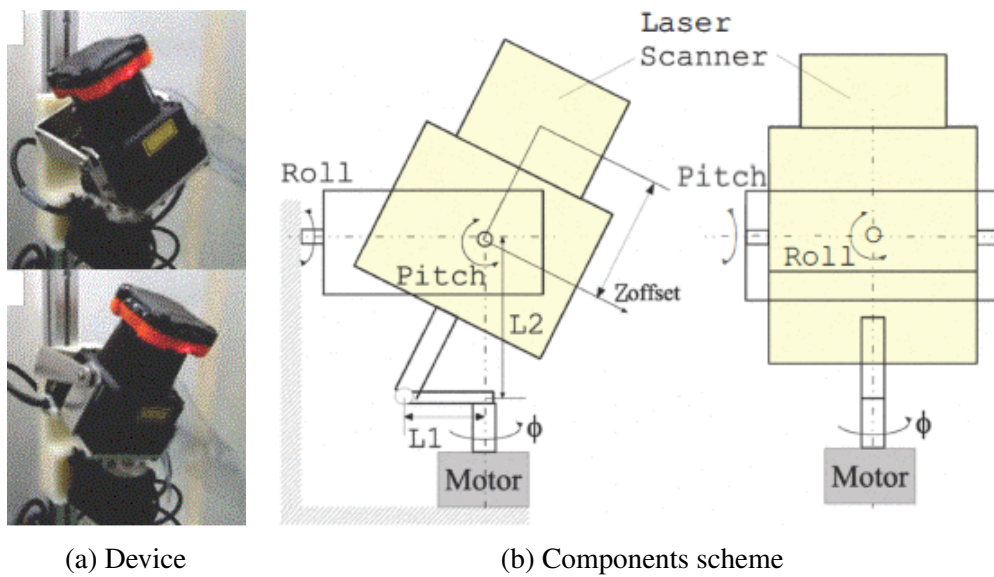


Fig. 2.6 Customized lidar system based on a 2D LRF with two new rotations [121]

the ground in a 2D grid and evaluate each cell based on the heights of the inner points [45] or the slope and flatness of those inner points [69].

## 2.4 Customized 3D lidar sensors

There is a growing interest in portable and affordable 3D lidar systems for new applications that require accurate and speedy range data acquisition, such as progress tracking in construction sites [74], precision agriculture [118], medical imaging [20], intelligent surveillance [9], or textile tailoring [110]. An alternative to high-end terrestrial scanners, like those used for digital terrain modeling or forest inventory [13], is to obtain dense 3D data by combining multiple views from less expensive sensors. Arbitrary motion can be produced manually [110][43] or by aerial [13] and ground vehicles [27], which requires registration processing. Nevertheless, the most common approach to build compact 3D devices from low-cost sensors has been to mount a 2D rangefinder onto a one degree-of-freedom (DOF) rotation mechanism that produces either a tilting or a spinning motion. Many examples of this type of rotating single-beam lidar (RSBL), mainly from the robotics community, can be found in the literature (see Table 2.1).

In the last few years, MBL rangefinders, such as those developed by Velodyne (Morgan Hill, CA, United States), are becoming increasingly popular and affordable. These sensors can be considered as a hybrid between 2D and 3D scanners, as they consist on a spinning structure that holds a number of independent laser transceivers to scan planes with different

Table 2.1 Representative examples of customized lidar systems based on a commercial device with an extra DOF.

	Type	Device	Major application
Batavia 2002 [6]	RSBL	Sick	Obstacle detection
Wulf 2003 [114]	RSBL	Sick LMS200	Density analysis
Weingarten 2006 [113]	RSBL	(2) Sick LMS200	Indoor environment reconstruction
Dias 2006 [24]	RSBL	Sick LMS200	Device comparison
Sheh 2006 [100]	RSBL	Hokuyo URG-04LX	Sensor configuration analysis
Ueda 2006 [106]	RSBL	Hokuyo URG-04LX	Mapping
Yoshida 2010 [121]	RSBL	Hokuyo UTM-30LX	Mapping
Morales 2011 [63] [64]	RSBL	Hokuyo UTM-30LX	Mapping and environment modeling
Xiao 2013 [115]	RSBL	Hokuyo UTM-30LX	Indoor mobile robot
Neumann 2014 [72]	RMBL	Velodyne HDL-64E	Underground mapping
Alismail 2015 [1]	RSBL	Hokuyo UTM-30LX-EX	Calibration for 3D mapping
An 2015 [3]	RSBL	Hokuyo URG-30LX	Plane extraction from indoor robot
Martinez 2015 [59]	RSBL	Hokuyo UTM-30LX-EX	UGV and UAV environment modeling
Özbay 2015 [76]	RSBL	Hokuyo UTM-30LX	UGV obstacle modeling
Moon 2015 [62]	RSBL	SICK LMS511-pro	Cargo ship modeling
Shaukat 2016 [99]	RSBL	Hokuyo UTM-30LX	RGB-D terrain modelling
Schubert 2016 [96]	RSBL	Hokuyo UTM-30LX	Robot mapping
Leingartner 2016 [50]	RMBL	Velodyne HDL-64E	Mapping
Neumann 2016 [71]	RSBL	Hokuyo UTM-30LX-EW	RMBL and MBL comparison
	RMBL	and Velodyne VLP-16	
Droeschel 2017 [27]	RSBL	Hokuyo UTM-30LX-EW	Robot mapping
Klamt 2017 [44]	RMBL	Velodyne VLP-16	Robot mapping
Pfrunder 2017[82]	RMBL	Velodyne VLP-16	Robot mapping

elevation angles within a fixed vertical field of view (FOV). For instance, the VLP-16 (or Puck) [108] is a compact and lightweight device (less than 1 kg) that scans 16 planes with an extended range (up to 100 m) and a large data rate (300,000 points/s) at a cost that is relatively low in comparison with other off-the-shelf sensors with 3D characteristics. These features have favored the adoption of MBLs in mobile applications, where dynamic point cloud registration along the vehicle's path compensates for device limitations in vertical resolution and FOV [111]. Nevertheless, a MBL has to be rotated in order to produce a complete spherical FOV [71].

In this context, rotating multi-beam lidars (RMBL) built by adding a DOF to a commercial MBL may arguably become a common solution to obtain affordable rapid full-3D high resolution scans in the near future. Some recent examples are summarized in Table 2.1. For instance, the rotating 2D Hokuyo UTM-30LX-EW (Osaka, Japan) scanner used by the Momaro robot in the DARPA Robotics Challenge of 2015 [27] has recently been replaced by a rotating Velodyne VLP-16 [44]. Moreover, Neumann *et al.* [72] built an RMBL based on a high-end 15 kg Velodyne HDL-64E to map underground mines from a wheeled robot. In a



Fig. 2.7 Unomotion, a customized lidar system based on a 2D LRF in horizontal layout with a complete rotation in roll [59]. This device extends from Unolaser [63] (see Appendix A.3).

later work, these authors have developed a rotating platform based on a Velodyne VLP-16 that also includes a 2D Hokuyo and other sensors [71].

The knowledge of scan density distribution is interesting to exploit the potential capabilities of a particular sensor/DOF combination for a given application [114][71]. To our knowledge, only a few works have explicitly addressed the analysis of 3D lidar data obtained by adding a rotation to a commercial rangefinder. Wulf and Wagner [114] analyzed the scanning patterns resulting from different arrangements of scan directions and rotation axes for a  $180^\circ$  2D Sick LMS200 (Waldkirch, Germany) scanner. This influential work studied the non-homogeneous distribution of range measurements of RSBLs by proposing a qualitative illustration of measured points on both a simulated hollow sphere surrounding the 3D scanner as well as on actual scans from representative environments. More recently, Alismail and Browning [1] used a synthetic hollow cuboid with the RSBL at its center for quantitative assessment of the scanning pattern for calibration purposes. Furthermore, Schubert et al. [96] aimed at optimizing the alignment of a 2D rangefinder with respect to the additional DOF. With this purpose, they claim that a cost function can be computed from the density distribution of points on the hollow sphere. Regarding RMBL, Neumann *et al.* [71] offered a comparison between several high-end MBL and customized RMBL devices using both a qualitative analysis of example scenes and quantitative performance indices that are repre-

sentative of particular device specifications, like scanning time, data rate, and average point density on the sphere.



UNIVERSIDAD  
DE MÁLAGA

# Chapter 3

## General framework for supervised learning classification of lidar point clouds based on voxel-based neighborhood

### 3.1 Introduction

Improving the effectiveness of spatial shape features classification from 3D lidar data is very relevant because it is largely used as a fundamental step towards higher level scene understanding challenges of autonomous vehicles and terrestrial robots. In this sense, computing neighborhood for points in dense scans becomes a costly process for both training and classification. This thesis addresses this problem by proposing a new general framework for implementing and comparing different supervised learning classifiers with a simple voxel-based neighborhood computation where points in each non-overlapping voxel in a regular grid are assigned to the same class by considering features within a support region defined by the voxel itself. The contribution provides offline training and online classification procedures as well as five alternative feature vector definitions based on principal component analysis for scatter, tubular and planar shapes. Moreover, the feasibility of this approach is evaluated by implementing a NN, as well as three other supervised learning classifiers found in scene processing methods: SVM, GP, and GMM (see section 2.2). A comparative performance analysis is presented using real point clouds from both natural and urban environments and two different 3D rangefinders (a tilting Hokuyo UTM-30LX and a Riegl).

The rest of the chapter is organized as follows. The next section proposes a general voxel-based neighborhood approach for supervised learning classification of spatial shape features. Section 3.3 describes the experimental setup and methodology for performance analysis offered in Section 3.4, which discusses results for different classifiers and feature vector definitions. The chapter is closed by the summary.

## 3.2 General voxel-based neighborhood framework for geometric pattern classification

This section proposes a voxel-based geometric pattern classification approach which can be generally used by supervised learning methods. General offline training and online classification procedures are detailed. Furthermore, data structures are proposed for the implementation of the point cloud and the input dataset.

### 3.2.1 Definitions

In general, classifiers produce a score to indicate the degree to which a pattern is a member of a class. For an input space with  $N$  patterns, the input dataset is defined as  $\mathcal{D} = \{(\mathbf{p}_i, t_i) | \forall i \in [1..N]\}$ , where  $\mathbf{p}_i = [p_i^1, \dots, p_i^{N_L}]$  is the  $i^{\text{th}}$  input pattern and  $t_i \in \mathcal{C}$  represents one of the  $N_C$  target classes, with  $\mathcal{C} = \{C_1, \dots, C_{N_C}\}$ . The  $N_L$  components of  $\mathbf{p}_i$  are computed according to a feature vector definition  $\mathcal{F}$ . Supervised learning needs a training dataset whose  $\mathbf{p}_i$  have been previously labeled with their corresponding  $t_i$ .

In this thesis, the goal is to classify scene points into three classes (i.e.,  $N_C = 3$ ):  $\mathcal{C} = \{C_1, C_2, C_3\}$ , where  $C_1$ ,  $C_2$ , and  $C_3$ , correspond to scatter, tubular and planar shapes, respectively. By using voxel-based neighborhood, all points within a voxel are assigned to the same class. With this aim, the point cloud in Cartesian coordinates is voxelized into a 3D grid of regular cubic voxels of edge  $E$ . Edge size depends on the scale of the spatial shapes to be detected in the point cloud. Only those voxels containing more points than a threshold  $\rho$  are considered to be significant for classification. Thus, the size  $N$  of the input dataset is the number of significant voxels.

### 3.2.2 Training and classification procedures

General training and classification procedures particularized for voxel-based neighborhood are shown in Fig. 3.1. Training is an offline process that has to be done once for a given classifier, whereas classification is performed online for each new point cloud. The training



procedure produces a multi-class classifier configuration consisting on a set of  $N_C$  classifiers that will be used in the classification procedure. Moreover, the choice of a feature vector definition and a particular classification method must be the same for the training and classification procedures.

A data structure  $V$  is defined to contain the input dataset  $\mathcal{D} = \{(\mathbf{p}_i, t_i)\}$ . When all  $t_i$  values in  $V$  have been set, either manually or automatically, this is considered a “classified  $V$ ”. An implementation of  $V$  is described in Section 3.2.4.

The training procedure (see Fig. 3.1a) uses a point cloud in Cartesian coordinates where the  $N_C$  geometric classes must be represented and discernible. After voxelization, the  $N$  significant voxels in the 3D grid are manually labeled with their corresponding class ( $t_i$ ) by a human supervisor. Then, a classified  $V$  data structure is built from the labeled voxels by computing  $\mathbf{p}_i$  for a particular choice of feature vector definition  $\mathcal{F}$  (e.g., one of the definitions proposed in Section 3.2.3). Training is performed for a given classification method with its particular parameters, where a different configuration is inferred for each class. The output of the training procedure is the trained classifier configuration.

The goal of the online classification procedure (see Fig. 3.1b) is to classify a new point cloud. The voxelized point cloud is used to create the  $V$  data structure with  $\mathbf{p}_i$  values computed

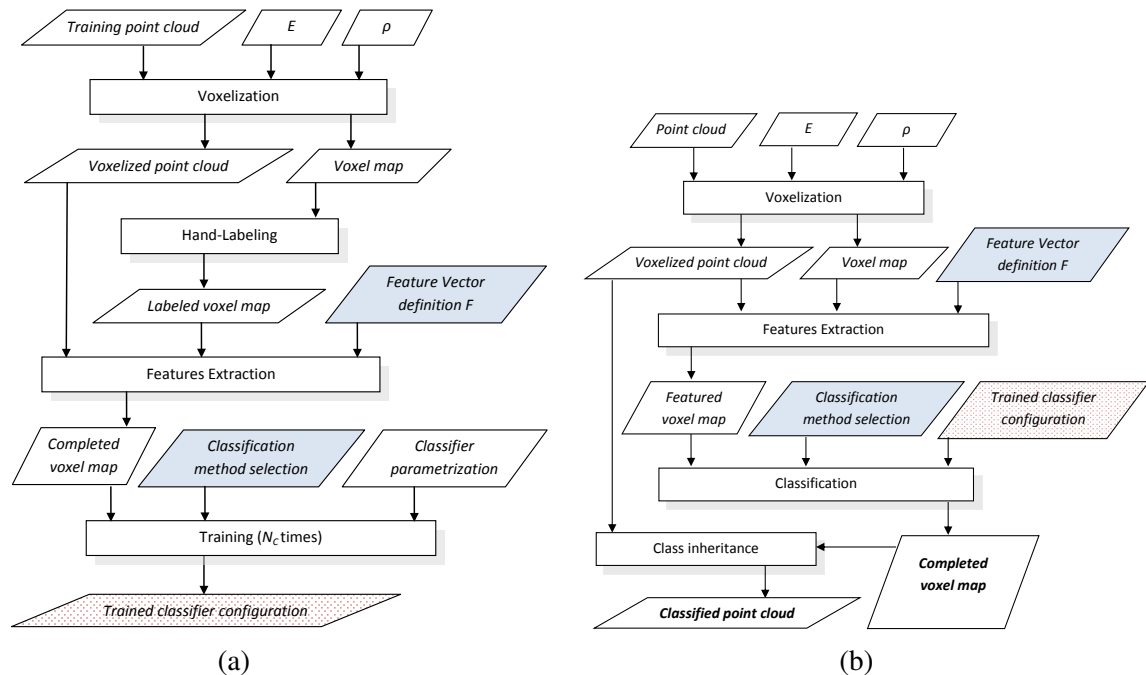


Fig. 3.1 Offline training (a) and online classification (b) procedures with voxel-based neighborhood computation. The choices of a feature definition and a classification method are common for both procedures (shaded in solid blue). The trained classifier configuration (shaded in dotted red) output in (a) is used in (b).

with the same feature vector definition as in the training procedure. In the classification step, the trained classifier configuration given by the training procedure completes the classified  $V$  by appending  $t_i$  values computed by considering the highest score of the  $N_C$  classifiers. With voxel-based neighborhood, the classification for each voxel is inherited by all points within its limits. Both can be considered as outputs for the method: *i*) the classified point cloud could be the output for algorithms that require point-based processing, and *ii*) the classified  $V$  can also be considered as a GFVM, as proposed in subsequent chapters of this thesis.

### 3.2.3 Extracting spatial shape features from voxels

The local spatial distribution of all the points within a voxel is obtained as a decomposition in the principal components of the covariance matrix from point Cartesian coordinates. These principal components or eigenvalues are sorted in ascending order as  $\lambda_0 \geq \lambda_1 \geq \lambda_2$  [47].

A feature vector  $\mathcal{F}$  consisting on a linear combination of the eigenvalues [47] and  $N_L = 3$  is generally considered in the literature [18][73]:

$$\mathcal{F} = [\lambda_0, \lambda_0 - \lambda_1, \lambda_1 - \lambda_2]. \quad (3.1)$$

This definition takes into account that scatterness has no dominant direction ( $\lambda_0 \approx \lambda_1 \approx \lambda_2$ ), tubularness shows alignment in one dominant direction ( $\lambda_0 \gg \lambda_1 \approx \lambda_2$ ), and planariness has two dominant directions ( $\lambda_0 \approx \lambda_1 \gg \lambda_2$ ).

Nevertheless, classifier convergence and performance can be affected by the definition and scaling of  $\mathcal{F}$  [105]. Thus, variants of Equation (3.1) based on the normalization and linear combination of eigenvalues could improve the performance of a particular classifier. Particularly, five feature vector definitions are considered in this thesis:

- $\mathcal{F}_1 = [\lambda_0, \lambda_1, \lambda_2]$ : eigenvalues from the covariance matrix.
- $\mathcal{F}_2 = [\lambda_0, \lambda_0 - \lambda_1, \lambda_1 - \lambda_2]$ : linear combination of the eigenvalues, as in Equation (3.1).
- $\mathcal{F}_3 = [\overline{\lambda_0}, \overline{\lambda_1}, \overline{\lambda_2}]$ : normalized eigenvalues.
- $\mathcal{F}_4 = [\overline{\lambda_0}, \overline{\lambda_0 - \lambda_1}, \overline{\lambda_1 - \lambda_2}]$ : normalization of the linear combination.
- $\mathcal{F}_5 = [\overline{\lambda_0}, \overline{\lambda_0 - \lambda_1}, \overline{\lambda_1 - \lambda_2}]$ : linear combination of normalized eigenvalues.

In  $\mathcal{F}_3$ ,  $\mathcal{F}_4$ , and  $\mathcal{F}_5$ , the overline over a value  $c$  denotes normalization of this value in  $[0,1]$  with respect to a 95 % confidence interval. This normalization is computed as follows:

$$\bar{c} = \frac{c - \min\{c_k\}}{\max\{c_k\} - \min\{c_k\}}, \quad \text{with } c \in \{c_k \mid k = 1..N_{95c}\}. \quad (3.2)$$

where  $N_{95c}$  represents the rounded integer number of the 95 % significant voxels in the middle of the distribution of  $c$ .

The input patterns  $\mathbf{p}_i$  in  $\mathcal{D}$  are computed by using the selected  $\mathcal{F}$  definition with the eigenvalues given by the covariance matrix corresponding to the points within the  $i^{\text{th}}$  significant voxel.

### 3.2.4 Data structures

In order to represent  $\mathcal{D}$ , the classification data structure  $V$  for the voxel map must be related to a list of Cartesian point cloud coordinates  $C$ . Particularly, efficient access to the list of points within each voxel is required both to compute the input patterns  $\mathbf{p}_i$  and to inherit classification by scan points. With this purpose, this section proposes two data structures that implement the voxel map  $V$  and the voxelized point cloud  $C$ . The voxel map  $V$  represents those voxels in a 3D spatial grid with constant size that have a representative number of points.

The structure  $V$  that implements  $\mathcal{D}$  is defined as a list of  $N$  elements, where the  $i^{\text{th}}$  element corresponds to a significant voxel and contains:

- $I_i \in [1..N_V]$ , the scalar index associated to the voxel. Assuming that a spatial 3D grid with  $N_V$  voxels includes all scan points, then a unique natural number  $I \in 1..N_V$  can be associated to each voxel [58]. This index is associated to the voxel and its inner points in the voxelization process. Since the whole 3D grid is a voxelized bounding box of the whole 3D scene, not all the voxels contains points, then  $N_V \geq N$  and usually  $N_V \gg N$ .
- $\mathbf{p}_i$ , feature vector values to be used as input pattern. It is computed based on the position of the inner points.
- $t_i \in [1..N_C]$ , a natural number representing the target class. This value is hand labeled in the training process and is the resulting class in the classification process.

Then,  $C$  is defined as a sorted list of all the  $m$  scan points from the original point cloud, where the  $j^{\text{th}}$  element has the following data:

- $(x_j, y_j, z_j)$ , the Cartesian point coordinates.

- $I_j \in [1..N_V]$ , a scalar index of the voxel that contains the point. This value is not unique in  $C$ , actually, will be massively repeated because it is shared by all the points belonging to one voxel.
- $t_j \in [1..N_C]$ , a natural number representing the target class. This value is inherited from  $V$ .

The computation of these data structures is as follows. First, considering the minimum bounding box for the point cloud, the space is split in a 3D grid composed by voxels. Then, all scan points in  $C$  are indexed with their corresponding voxel index, which is also used to sort the list. After that, if there are more than  $\rho$  consecutive elements in  $C$  with the same index number, then a new entry for that voxel is created in  $V$ . After voxel classification, points in  $C$  with the same voxel index inherit the target class of the corresponding voxel in  $V$ . Points in non-significant voxels will remain unclassified (i.e., with a null value in the target class field).

### 3.3 Experimental setup and methodology

This section describes the training and evaluation datasets, the parametrization of classifiers, and the methodology used for the comparative performance analysis offered in Section 3.4.

#### 3.3.1 Experimental datasets

Classification has been applied to three evaluation point clouds obtained with representative sensors and illustrative of natural and urban environments:

- *Urban*. This point cloud of a urban environment is a subset of the Sydney Campus dataset [26], which was scanned by a Riegl sensor (see Fig. 3.2). This is a complex scene which involves structured objects, mostly planes, such as buildings and flat floors.
- *Natural\_1* and *Natural\_2*. These point clouds are dominated by unstructured objects such as bushes, trees and rough terrain. Both scenes were scanned on natural areas close to Universidad de Málaga by a UNOLaser rangefinder. This sensor is based on pitching a 2D Hokuyo UTM-30LX [64], with a maximum range of 30 m and horizontal and vertical fields of view of  $270^\circ$  and  $131^\circ$ , respectively. The first scan is from a complex scene with dense tree crowns (see Fig. 3.3) and the second includes both bushes and tall trees with visible trunks (see Fig. 3.4).

As for the training procedure, a different point cloud has been considered:

- *Garden*. This point cloud contains elements from a semi-structured environment where the three geometric classes can be discernible for hand labeling: planar floor, tubular tree trunks, and scattered tree crowns (see Fig. 3.5). This scene was scanned with the UNOLaser sensor.

Evaluation and training point clouds have been voxelized with  $E = 0.5$  m and  $\rho = 10$  (see Section 3.2.1), which were empirically determined [84]. Table 3.1 summarizes voxelization and hand labeling of experimental point clouds (evaluation datasets have also been hand labeled to evaluate classification performance). The table presents the resulting number of voxels and points included in the corresponding  $V$  structures, as well as the percentage of voxels for each class after hand labeling. In the *Urban* dataset, most voxels have been labeled as planar because clear floor and building walls dominate the scene. Conversely, in the *Natural\_1* and *Natural\_1* voxelized point clouds a majority of the voxels are scatter or tubular due to bushes and trunks and treetops.

### 3.3.2 Classifiers parametrization

The parametrization of the SVM classifier is the following:

- Function kernel: radial basis function as in [38].

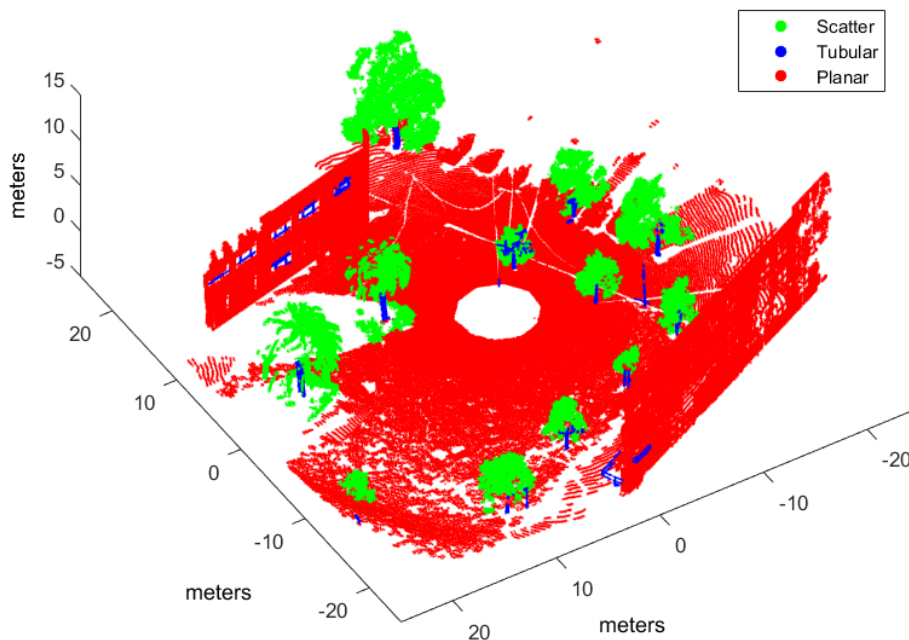
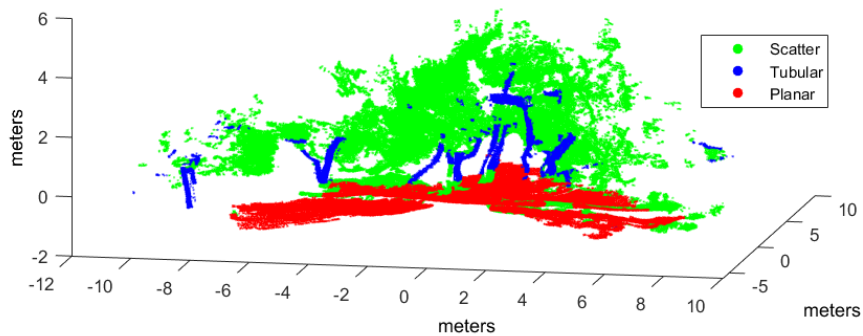
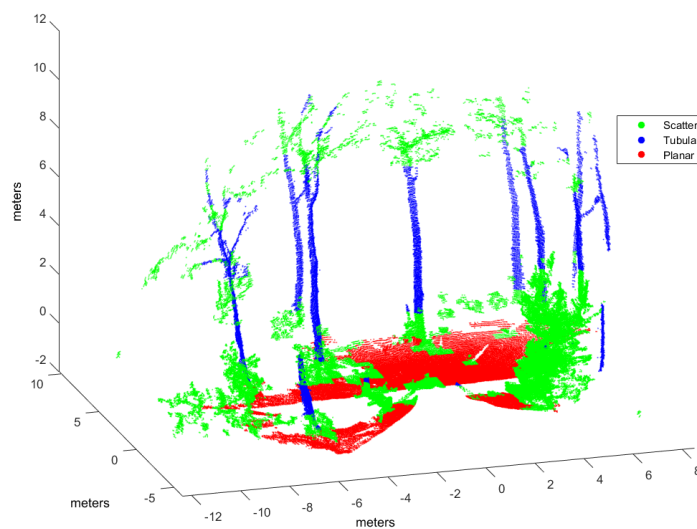
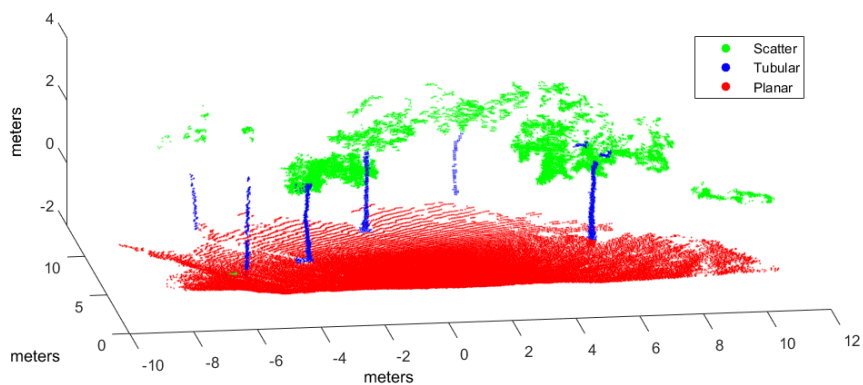


Fig. 3.2 Hand labeled *Urban* point cloud.

Fig. 3.3 Hand labeled *Natural\_1* point cloud.Fig. 3.4 Hand labeled *Natural\_2* point cloud.Fig. 3.5 Hand labeled *Garden* point cloud.

- Box constraint: infinite.

The parameters used for the GP classifier are:

Table 3.1 Characteristics of hand labeled voxels of experimental point clouds

Dataset type	Dataset	#Voxels	#Points	Voxels percentage		
				Scatter	Tubular	Planar
Evaluation	<i>Urban</i>	13713	1473757	27.7	16.3	55.9
	<i>Natural_1</i>	1877	618913	72.8	6.2	20.9
	<i>Natural_2</i>	1346	267514	58.6	61.6	17.8
Training	<i>Garden</i>	974	128836	34.9	4.2	60.9

- Number of function evaluations: 30.
- Covariance function: squared exponential function with automatic relevance determination.
- Mean function: constant mean function.
- Inference method: expectation propagation algorithm.
- Likelihood function: cumulative Gaussian function.
- Hyperparameters of mean and covariance: 0 and (1,1,1,1) respectively (i.e., all length-scales and the signal magnitude are initialized to 1 and represented in the log space).

In GMM, the parameters are:

- $n_g^{scatter} = n_g^{tubular} = n_g^{planar} = 10$ .
- Marginal likelihood maximization: EM algorithm [11].

The proposed NN based classifier uses the following configuration:

- Network topology: multi-layer perceptron with one hidden layer of 100 neurons (this number was determined using a cascade learning constructive processing in which neurons are added to the hidden layer, one at time, until there is no further improvement in network performance).
- Transfer function: logistic transfer functions in hidden and output layers.
- Back-propagation learning algorithm: Levenberg-Marquardt.
- Learning constant: 0.02.

Besides, the training process of the NN must be stopped at an appropriate iteration to avoid overfitting. This iteration is found by the early stopping method of training [2], in which the training dataset is split into an estimation subset (80 % of the training set) and a validation subset (the remaining 20 %). More details of the configuration and implementation of the NN classifier can be found in [85].

### 3.3.3 Methodology

The performance of the classifiers will be compared by using classification statistical measures for each class. In particular, confusion matrices along with a multi-class extension of Matthew's Correlation Coefficient (*MCC*) have been considered.

In a classification problem with  $N_C$  target classes, a confusion matrix is the square matrix  $M$  ( $N_C \times N_C$ ) whose  $ij^{th}$  entry,  $M_{ij}$ , is the number of elements of true class  $i$  that have been assigned to class  $j$  by the classifier [42]. Therefore, an ideal classifier would yield a diagonal  $M$ . In this case, elements are points from significant voxels. Furthermore, in order to achieve a clear comparison between different datasets, normalized confusion matrices can be defined. Elements in the are defined as:

$$\overline{M}_{ij} = \frac{M_{ij}}{\sum_{i=1}^{N_C} M_{ij}} \times 100, \quad (3.3)$$

where the sum of row elements is 100.

The generalization of *MCC* for the multi-class problem [36] is used as a reference performance measure on unbalanced datasets [42], which can be defined as follows:

$$MCC = \frac{\sum_{i,j,k=1}^{N_C} M_{ii}M_{kj} - M_{ji}M_{ik}}{\sqrt{\sum_{i=1}^{N_C} \left[ \left( \sum_{j=1}^{N_C} M_{ji} \right) \left( \sum_{m,n=1; m \neq i}^{N_C} M_{nm} \right) \right]} \sqrt{\sum_{i=1}^{N_C} \left[ \left( \sum_{j=1}^{N_C} M_{ij} \right) \left( \sum_{m,n=1; m \neq i}^{N_C} M_{mn} \right) \right]}}. \quad (3.4)$$

*MCC* summarizes the confusion matrix into a single value in the [-1,1] range, where 1 represents a perfect classification and -1 extreme misclassification.



Table 3.2 Performance of classifiers ( $MCC$  and  $\bar{M}$ ) using feature vector definition  $\mathcal{F}_2$  for the evaluation datasets.

	Natural_1			Natural_2			Urban					
	$MCC$	$\bar{M}$			$MCC$	$\bar{M}$			$MCC$	$\bar{M}$		
GMM ( $\mathcal{F}_2$ )	0.4631	74.3	18.7	7.0	0.5696	72.8	18.1	9.1	0.4962	81.5	11.7	6.8
		55.6	39.3	5.1		18.3	65.4	16.3		31.1	43.4	25.5
		14.2	9.2	76.6		15.8	6.6	77.6		11.9	15.0	73.1
GP ( $\mathcal{F}_2$ )	0.0958	0.0	3.7	96.3	0.0005	0.0	1.3	98.7	0.0486	0.0	4.2	95.8
		0.0	6.1	93.9		0.0	0.0	100		0.0	2.9	97.1
		0.0	0.2	99.8		0.0	0.0	100		0.0	0.3	99.7
NN( $\mathcal{F}_2$ )	0.6461	91.7	1.8	6.5	0.7927	95.9	2.3	1.8	0.6557	95.9	0.2	4.0
		56.4	37.6	6.0		28.7	68.2	3.1		11.7	22.9	65.4
		4.9	2.1	93.0		6.9	1.0	92.1		0.0	0.0	100
SVM( $\mathcal{F}_2$ )	0.3091	48.6	0.0	51.4	0.1164	9.4	0.0	90.6	0.3152	43.2	0.0	56.8
		32.7	0.0	67.3		2.2	0.0	97.8		14.7	0.0	85.3
		0.8	0.0	99.2		1.3	0.0	98.7		0.1	0.0	99.9

### 3.4 Performance analysis and comparison

This section discusses experimental results where the voxel-based approach proposed in Section 3.2 has been applied to the NN classifier and other supervised learning classifiers: SVM, GP, and GMM. First, all classifiers are compared with a representative feature vector definition. Then, an experimental analysis is performed to select an appropriate feature vector definition for each classifier. The section also includes a discussion of computation times as well as a comparison with a point-wise neighborhood classifier.

#### 3.4.1 Performance evaluation with linear combination of eigenvalues

The evaluation datasets described in Section 3.3.1 have been used to compare the performance of the four classifiers trained with  $\mathcal{F}_2$ , which is the feature vector definition given by Lalonde *et al.* [47]. Table 3.2 presents  $MCC$  and  $\bar{M}$  for each classifier in all evaluations datasets. Regarding  $MCC$ , the NN classifier achieves the best results in all datasets. The GMM classifier obtains the second best performance whereas SVM and GP get poor results. Especially, SVM never classifies patterns as class  $C_2$  (tubular), as indicated by null values in the second column of  $\bar{M}$  for all datasets. Similarly, GP classifies most points (over 90 %) as class  $C_3$  (planar). These results indicate poor performance of  $\mathcal{F}_2$  for some classifiers.

#### 3.4.2 Performance evaluation with different feature vector definitions

This section offers an experimental analysis to find a suitable selection of  $\mathcal{F}$  for each classifier. With this purpose, all classifiers have been trained with the five feature vector definitions

described in Section 3.2.3 using the *Garden* dataset. Table 3.3 summarizes this analysis by showing the corresponding *MCC* values. These results indicate that GP and SVM are strongly affected by the choice of the feature vector while GMM offers good results for all definitions. In this sense, the NN method achieves better results with the non-normalized definitions, which can be explained by the nonlinear qualities of the MLP. All in all, the best scores have been obtained with  $\mathcal{F}_2$  for NN,  $\mathcal{F}_4$  for GMM and GP and  $\mathcal{F}_5$  for SVM. These definitions have been selected as the most appropriate choice for each classifier.

Comparative results with the corresponding  $\mathcal{F}$  selections are given in Table 3.4. Regarding *MCC*, the NN classifier maintains the best results in all datasets. Besides, GP becomes the second best, clearly improving with respect to Table 3.2 (where it obtained the worst performance), which denotes the importance of an appropriate selection of  $\mathcal{F}$ . As for  $\bar{M}$ , it can be noted that class  $C_2$  (tubular) is the most difficult to classify (as indicated by low values in the diagonal element of the second row). In this difficult class, NN consistently outperforms all other classifiers, and reaches 68.2% of true positives in the *Natural\_2* dataset.

Figs. 3.6-3.8 illustrate the application of our NN classifier with the voxel-based neighborhood approach for the three evaluation datasets. These classification results show good accordance with the ground truth (i.e., hand labeled) values given in Figs. 3.2-3.4.

### 3.4.3 Computation time

Table 3.5 presents execution times corresponding to a Matlab implementation of the classifiers running on a Core i7 processor with a clock frequency of 3.7 Ghz and 16 Gb of RAM. Computation of data structure  $V$  is common for all classifiers. Then, total computation time is obtained by adding the time for  $V$  computation to the training process time (in the offline procedure) or to the classification process time (in the online procedure).

$V$  computation time includes voxelization as well as calculation of covariance matrices and their associated eigenvalues for every voxel. This value is proportional to the number of voxels in the data structure, which is greater for the *Urban* dataset (see Table 3.1).

Table 3.3 Performance of classifiers (*MCC*) for the training dataset with different definitions of the feature vector  $\mathcal{F}$ .

	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_4$	$\mathcal{F}_5$
GMM	0.8667	0.8725	0.7856	0.9562	0.7988
GP	0.3414	0.0042	0.5996	0.8224	0.7420
NN	0.8295	0.8384	0.5659	0.4240	0.5633
SVM	0.0723	0.1687	0.4722	0.3275	0.5268

Table 3.5 shows that GP requires much more computation time, for both training and classification, than the rest of classifiers. For offline training, the times for the training process, which offer considerable differences between the four classifiers, are greater than the time required for  $V$  computation. As for online classification, GMM, NN and SVM achieve classification times that are significantly faster than  $V$  computation, so their total computation

Table 3.4 Performance of classifiers ( $MCC$  and  $\bar{M}$ ) for the evaluation datasets using selected feature vector definition ( $\mathcal{F}_2$  for NN,  $\mathcal{F}_3$  for SVM and  $\mathcal{F}_4$  for GMM and GP).

	<i>Natural_1</i>			<i>Natural_2</i>			<i>Urban</i>					
	$MCC$	$\bar{M}$			$MCC$	$\bar{M}$			$MCC$	$\bar{M}$		
GMM ( $\mathcal{F}_4$ )	0.5352	94.4	2.8	2.8	0.5756	93.9	4.1	2.0	0.6021	97.4	1.3	1.3
		81.2	12.7	6.1		66.7	27.4	5.9		32.6	14.1	53.3
		13.1	1.1	85.8		8.8	2.8	88.4		2.6	0	97.4
GP ( $\mathcal{F}_4$ )	0.5382	93.9	1.2	4.9	0.5871	96.3	0.8	2.9	0.6384	98.8	0.1	1.1
		64.2	9.6	26.2		32.0	23.0	45.0		13.6	15.3	71.1
		6.9	1.0	92.1		7.4	2.7	89.9		0.5	0	99.5
NN( $\mathcal{F}_2$ )	0.6461	91.7	1.8	6.5	0.7927	95.9	2.3	1.8	0.6557	95.9	0.2	4.0
		56.4	37.6	6.0		28.7	68.2	3.1		11.7	22.9	65.4
		4.9	2.1	93.0		6.9	1.0	92.1		0.0	0.0	100
SVM ( $\mathcal{F}_5$ )	0.4483	83.6	0.6	15.8	0.4646	86.9	1.3	11.8	0.5082	84.3	1.2	14.5
		54.7	0.8	44.5		39.6	0.3	60.1		20.7	0.7	78.6
		6.4	0.1	93.5		6.8	0.0	93.2		0.3	0.0	99.7

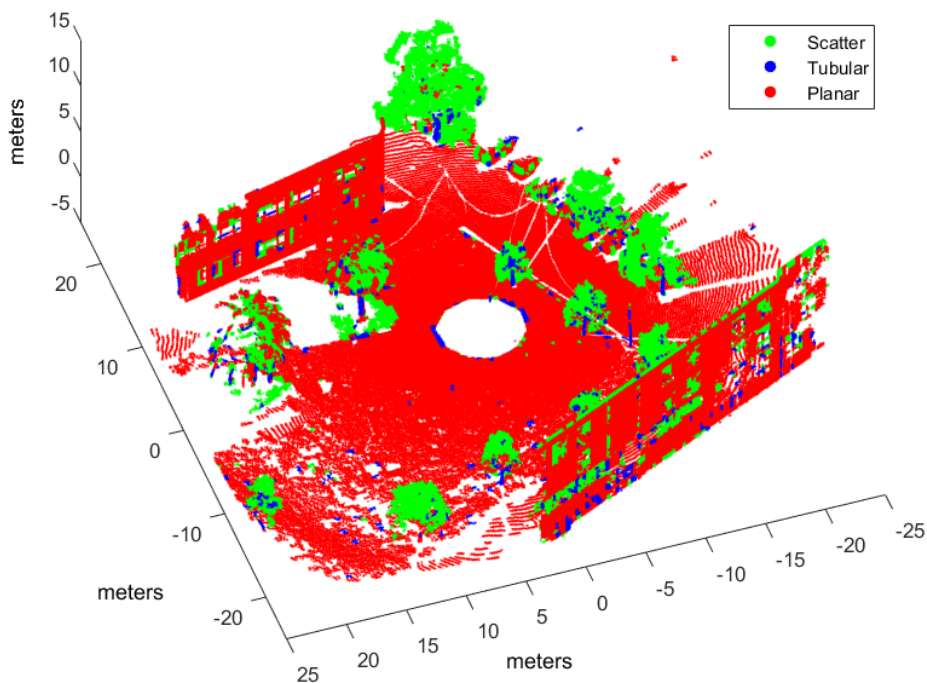


Fig. 3.6 *Urban* point cloud classified by NN with  $\mathcal{F}_2$ .

Table 3.5 Computation times for training and classification, in seconds.

	Training		Classification	
	<i>Garden</i>	<i>Urban</i>	<i>Natural_1</i>	<i>Natural_2</i>
<i>V computation</i>	0.154	1.895	0.432	0.263
GMM	0.557	0.018	0.004	0.003
GP	104.673	3.995	4.010	3.795
NN	5.790	0.121	0.035	0.032
SVM	43.652	0.112	0.017	0.018

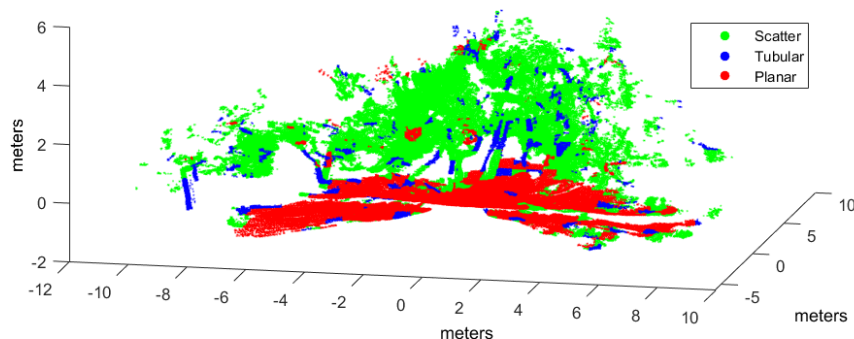
times are similar and close to that value. Since the best classification performance in Table 3.4 was achieved by NN, it can be concluded that NN accomplishes an outstanding compromise between performance and computation time.

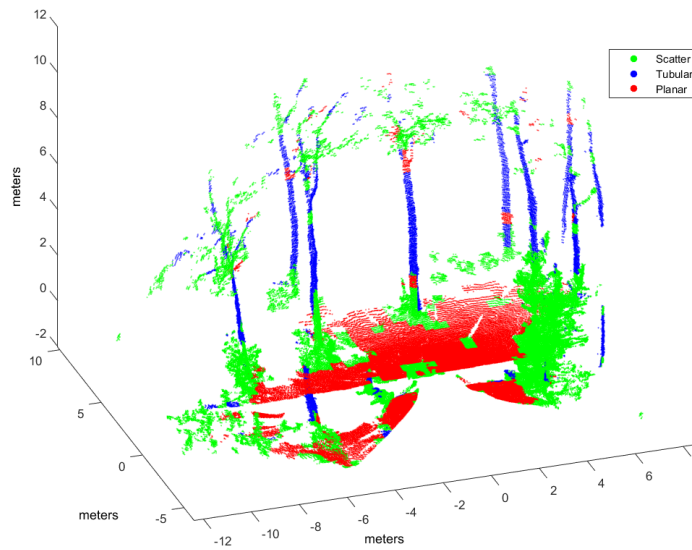
### 3.4.4 Comparison with point-wise neighborhood classification

Performance of voxel-based neighborhood has also been compared against point-wise neighborhood. In particular, the experimental datasets have been processed with a point-wise GMM classifier with  $\mathcal{F}_2$  (i.e., the configuration used by Lalonde *et al.* [47]) with a support region defined by a radius of 0.5 m. Classification performance and computation times are presented in Tables 3.6 and 3.7, respectively.

Regarding classification performance, Table 3.6 presents  $MCC$  and  $\bar{M}$  for point-wise GMM in all evaluation datasets. Comparing  $MCC$  Table 3.6 against the first row of Table 3.4, it can be appreciated that performance results are very similar. Particularly, voxel-based neighborhood outcores the point-wise method in the *Natural\_2* and *Urban* datasets.

Total computation time is the sum of neighborhood computation and training/classification times, which are given as two separate rows in Table 3.7. In this case, most of the time is

Fig. 3.7 *Natural\_1* point cloud classified by NN with  $\mathcal{F}_2$ .

Fig. 3.8 *Natural\_2* point cloud classified by NN with  $\mathcal{F}_2$ .Table 3.6 Performance of GMM classifier with point-wise neighborhood using feature vector definition  $\mathcal{F}_2$  for the evaluation datasets.

	<i>Natural_1</i>			<i>Natural_2</i>			<i>Urban</i>					
	<i>MCC</i>	$\bar{M}$		<i>MCC</i>	$\bar{M}$		<i>MCC</i>	$\bar{M}$				
Point – wise GMM	0.5392	79.2	7.6	13.2	0.5288	80.1	7.1	12.8	0.5797	79.4	6.6	14.0
		47.0	32.8	20.2		16.7	33.3	50.0		23.2	35.3	41.5
		6.9	1.9	91.2		8.5	4.1	87.4		3.3	1.2	95.5

Table 3.7 Computation times for point-wise neighborhood training and classification, in seconds.

	Training		Classification	
	<i>Garden</i>	<i>Urban</i>	<i>Natural_1</i>	<i>Natural_2</i>
<i>Neighborhood computation</i>	63.12	933.73	574.98	166.28
Point-wise GMM	8.89	9.29	1.36	0.63

used for neighborhood computation. The comparison of this table with Table 3.5 shows that computation times for voxel-based neighborhood are dramatically reduced with respect to point-wise neighborhood.

In general, these results indicate that voxel-based neighborhood classification achieves a dramatic improvement in computation time with respect to point-wise neighborhood, while no relevant differences in performance can be appreciated. Furthermore, voxel-based NN has accomplished better classification performance with the experimental datasets.

### 3.5 Summary

Many point cloud classification problems targeting real-time applications such as autonomous vehicles and terrestrial robots have received attention in recent years. Among these problems, improving the effectiveness of spatial shape features classification from 3D lidar data remains a relevant challenge because it is largely used as a fundamental step towards higher level scene understanding solutions. In particular, searching for neighboring points in dense scans introduces a computational overhead for both training and classification.

The originality of this thesis is a new general framework for supervised learning classifiers to reduce the computational load based on a simple voxel-based neighborhood definition where points in each non-overlapping voxel of a regular grid are assigned to the same class by considering features within a support region defined by the voxel itself. The contribution comprises offline training and online classification procedures as well as five alternative feature vector definitions based on principal component analysis for scatter, tubular and planar shapes.

Moreover, the feasibility of this approach has been evaluated by implementing four types of supervised learning classifiers found in scene processing methods: our NN model, support vector machines (SVM), Gaussian processes (GP), and Gaussian mixture models (GMM). An experimental performance analysis has been carried out using real scans from both natural and urban environments and two different 3D rangefinders: a tilting Hokuyo and a Riegl. The major conclusion from this analysis is that voxel-based neighborhood classification greatly improves computation time with respect to point-wise neighborhood, while no relevant differences in scene classification accuracy have been appreciated. Results have also shown that the choice of suitable features can have a dramatic effect on the performance of classification approaches. All in all, classification performance metrics and processing time measurements have confirmed the benefits of the NN classifier and the feasibility of the voxel-based neighborhood approach for terrestrial lidar scenes.

One additional advantage of processing each non-overlapping cell by using points from only that same cell is that this favors parallelization [94]. Developing a parallel version of the proposed method to improve online classification time with multi-core computers will be addressed in future work. Furthermore, it will be also interesting to adapt the method for incremental update of classification results with consecutive scans.

As a result, a GFVM is obtained where each voxel is labelled as one of these geometric classes: scatter, linear or flat surface. Fig. 3.9 shows a dense point cloud modelled as a GFVM. The scene shows a natural environment in Malaga captured by the UNOLaser rangefinder. The geometric classes associated to each voxel can be applied to identify scene objects such as ground, tree trunks and vegetation, which will be addressed in Chapter 4.

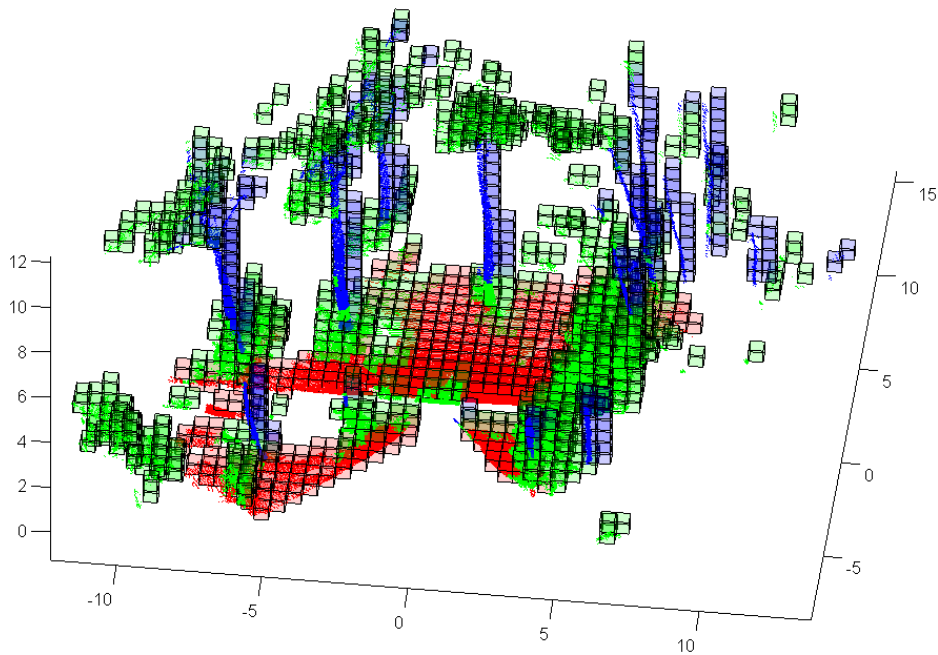


Fig. 3.9 Geometrically featured voxel map from a natural environment. Red voxels have been classified as planar, blue as tubular, and green as scatter.

The scientific production of this chapter is listed below:

- [85] IWANN' 15, 13th International Work-Conference on Artificial Neural Networks. Multi-layer Perceptrons for Voxel-based Classification of Point Clouds from Natural Environments. 2015. (Core B)
- [88] Sensors. Voxel-Based Neighborhood for Spatial Shape Pattern Classification of Lidar Point Clouds with Supervised Learning. 2017. (Journal Q1)



UNIVERSIDAD  
DE MÁLAGA



# Chapter 4

## Segmentation and navigability assessment based on geometrically featured voxel maps

### 4.1 Introduction

Autonomous navigation for mobile robots in complex 3D environments requires efficient characterization of the perceived scene structure. This chapter proposes novel contributions for point cloud segmentation based on the geometrically featured voxel maps presented in Chapter 3. In particular, the proposed solution involves two steps (see Fig. 4.1): *i*) a ground segmentation method specially designed for natural environments which serves as a basis for subsequent isolation of individual objects, and *ii*) the generation of a traversability map based on occupancy grids which can be suitable for mobile robots in natural environments.

As in [26], ground segmentation is considered in the first place, which favors posterior object segmentation and navigability assessment. Thus, the ground segmentation algorithm is the cornerstone of the proposed solution for object segmentation and navigability assessment. Other authors have approached ground segmentation by fitting a flat plane [126] [125] or by

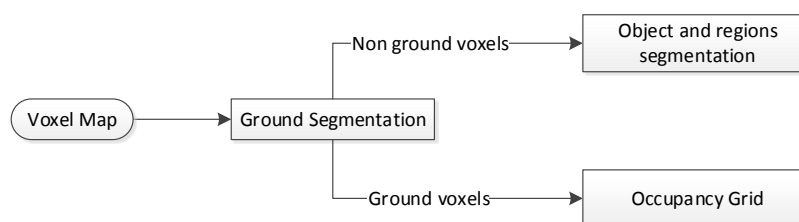


Fig. 4.1 General description of the segmentation and navigability assessment solution

using a threshold, for the height mean [122][53] or for both the height mean and the vertical variance [26]. These approaches are useful for urban environments, but they do not consider the 3D complexity of natural terrain. [112] combines both approaches in an adaptive ground segmentation method with good results in uneven and sparse regions, however it is based in a data representation which is only affordable for 3D point cloud obtained by multibeam sensors, such as Velodyne. This chapter proposes a new 3D ground segmentation algorithm for dense point clouds specially designed for natural environments where big slopes, non-flat areas, and isolated areas have been considered. A number of vertically aligned voxels can be considered as ground points in order to properly define slopes, mounds, or other terrain irregularities. To identify these ground voxels, the ground segmentation algorithm consists of a sequence of refining rules. An appropriate sequence for the rules has been determined from experimental analysis. Furthermore, the influence of voxel size has also been analyzed. Besides, ground segmentation is used to extend the voxel map representation by adding a new attribute to the voxels, apart from their geometric class.

After ground segmentation, objects can be individually isolated and geometrically featured by using the class of their corresponding voxels. To this end, a two-step object segmentation is proposed. First, the non-ground points are divided into individual objects by merging adjacent voxels, and, secondly, each resulting object is split into regions with the same geometric class by merging adjacent voxels with same class.

Besides, this chapter presents a new navigability assessment method for ground vehicles on natural terrain that is also based on ground segmentation. Navigability assessment can be roughly classified into three main categories: a binary classification problem, traversable or non-traversable terrain, a finer estimation with a continuous traversability score, or a classification with various classes. [78] reviews this terrain traversability analysis problem and distinguishes the geometry-based approaches according to the criteria employed for estimating traversability. A discrete model of occupancy or traversability has been adopted by other authors by evaluating only the heights [45] or the slope and flatness of each area [70]. The solution proposed in this chapter generates a 2D occupancy grid which represents accessible areas by considering the geometric attributes and the height of each voxel over the ground level in relation to vehicle dimensions.

All in all, the methods presented in this chapter contribute to the characterization of 3D terrain and complex objects in the environment. The method extends applicability to point cloud in unstructured environments because of two reasons: *i*) geometric features are identified regardless of their orientation and *ii*) ground segmentation considers sharp slopes, non-flat areas, and isolated areas. The aim of this study is to provide a novel solution for 3D

segmentation and navigability assessment that can facilitate tasks such as object recognition and path planning in natural environments for autonomous mobile robots.

The rest of the chapter is organized as follows. Section 4.2 describes the steps of the proposed segmentation algorithm and the subsection 4.2.3 explains the experiments and metrics to evaluate the algorithm and shows the results obtained. Section 4.3 proposes the occupancy grid generation method and contains a subsection which shows several experiments performed. Finally, conclusions are presented.

## 4.2 Segmentation methods for unstructured environments

This section presents segmentation methods that are based on a GFVM voxel structure, as described in Chapter 3. Particularly, the proposed segmentation method is divided into two major stages, as seen in the upper path of Fig. 4.1. First, the ground is extracted from the scene by defining a set of rules specifically considered for unstructured environments. After that, non ground voxels are used to segment objects, which are subsequently split into regions with the same geometric classification. Furthermore, these segmentation algorithms are used to add new attributes to the voxel map.

### 4.2.1 Ground segmentation

Some works have tested the benefit of using the ground as a separator between objects [26]. Generally, ground segmentation methods assume that the ground as a whole is continuous, almost flat, or the largest object in the scene. These assumptions are not necessarily true in natural environments where the ground can have different types of discontinuities due to

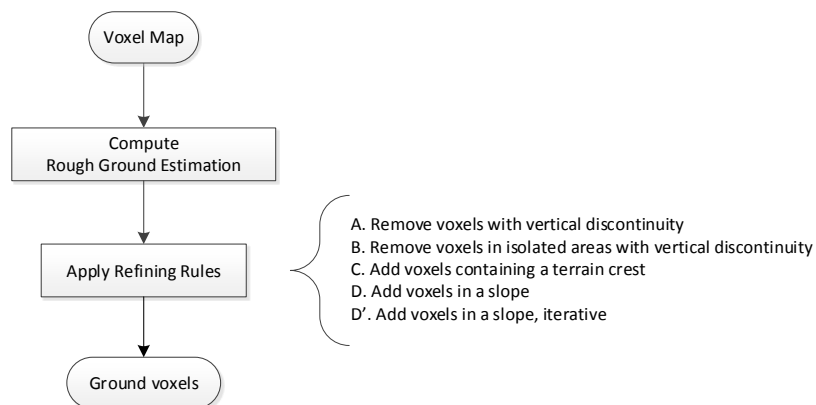


Fig. 4.2 Flow diagram of the ground segmentation algorithm.

sharp slopes, terrain roughness, or the presence of rocks or bushes. Besides, ground surface may not necessarily imply traversable areas for vehicles, as in urban environments.

This section presents a ground segmentation method specially designed for this kind of unstructured environments. The proposed algorithm, which identifies the set of voxels  $G$  that correspond to ground points, is illustrated in the flow diagram shown in Fig. 4.2. A planar  $XY$  grid with edge size  $E$  is defined on the base of the GFVM defined in Chapter 3. Thus, each  $XY$  cell may support a number of occupied voxels in the corresponding GFVM at different heights (coordinate  $Z$ ). Then, the algorithm starts from a rough estimation of  $G$  consisting of the set of the lowest occupied voxels, if any, over the  $XY$  grid.

Starting from this rough estimation, several rules are proposed to refine  $G$  by adding and removing voxels from GFVM by considering connectivity between occupied voxels. In this work, two voxels are considered as adjacent if they touch any of their facets, edges or corners. Then, two voxels are connected if there is an adjacency path between them. As each rule modifies  $G$ , the order in which the rules are performed is relevant, which will be addressed in Section 4.2.3. The proposed rules, which are illustrated in Fig. 4.3, are the following:

- *A. Remove voxels with vertical discontinuity.*

*A voxel is removed from  $G$  if it is not adjacent to any other voxel in  $G$  with different  $XY$  coordinates.*

This rule aims to remove voxels which are much further up than their neighbors (see Fig. 4.3(a)). With this purpose it is assumed that vertical discontinuities indicate non-ground objects even if sharp slopes can correspond to ground points. This rule implies that it is possible that not all  $XY$  cells support ground voxels.

- *B. Remove voxels in isolated areas with vertical discontinuity.*

Let us define  $G_l \in G$  such that  $G_l$  contains all the voxels connected to the lowest voxel in  $G$ . Then:

*All voxels from a connected set of voxels  $G_d \in G$  such that  $G_d \neq G_l$  are removed from  $G$  if the closest pair of voxels from  $G_l$  and  $G_d$  have a height difference greater than  $E$ .*

This rule, which is illustrated in Fig. 4.3 (b), assumes that the area with the lowest points corresponds to the ground [38]. Disconnected areas, as floating objects, are not considered as ground unless they have a similar height. Furthermore, terrain irregularity is accounted for by considering the actual heights between the closest pair of voxels instead of the lowest voxel value.

- *C. Add voxels containing a terrain crest.*

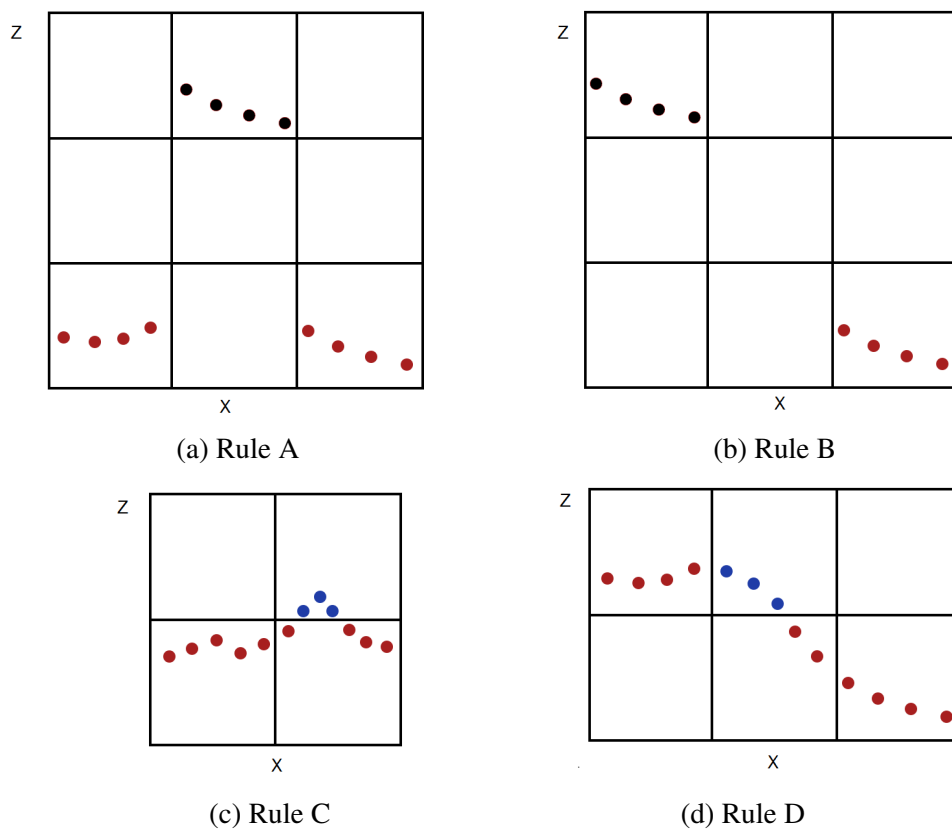


Fig. 4.3 Illustration of ground segmentation rules by considering GFVM voxels for a single  $Y$  value. Red points correspond to current ground voxels that are not modified; voxels with black points in (a) and (b) are removed from the set of ground voxels; and voxels with blue points in (c) and (d) are added to the set of ground voxels.

*A voxel from the GFVM is added to  $G$  if its upper neighbor is not occupied and the average height of its inner points is less than a fixed threshold and its lower neighbor is in  $G$ .*

As illustrated by Fig. 4.3(c), protruding ground relief may require more than one voxel in  $G$  over the same  $XY$  cell. However, if the upper voxel in GFVM is occupied or its inner points are high, then the voxel is not added to the ground, as it may be part of an object.

- *D. Add voxels in a slope.*

*A voxel from the GFVM is added to  $G$  if its lower neighbor and at least one adjacent voxel with the same height are in  $G$ .*

Rule *D*, which is illustrated in Fig. 4.3 (d), finds pairs of adjacent voxels in  $G$  with different  $XY$  coordinates and  $E$  height difference. This rule assumes these pair of voxels belongs to a slope in the terrain and the upper voxel of the lowest from the pair also belongs to the terrain  $G$ .

- *D'. Add voxels in a slope, iterative.*

*New voxels added to  $G$  with rule *D* may also belong to a new slope where rule *D* may be applied.*

This is an iterative alternative to rule *D*, where subsequent application of the rule can refine the slope in the terrain  $G$  until no new voxel is added.

## 4.2.2 Object segmentation

Once the ground has been extracted, the scene partition into objects is found by clustering together adjacent voxels [26]. In addition, each object is also split into regions which are obtained by merging adjacent voxels with same geometric class. For example, a tree object would be split into two regions at least: *i*) a linear region relative to the trunk and branches and *ii*) a scattered region relative to the tree crown. Figs. 4.4 and 4.5 show the segmentation in objects and regions respectively from the same natural scene.

This double segmentation may be helpful in subsequent applications, for instance scene description and/or recognition. Several statistical features such as width, height, depth, centroid position and distance from the ground, can be computed from each region to be used as descriptors or inputs in future object classification algorithms. For instance, an object with a scattered region at the top and a long linear region below and just over the ground, could be identified as a tree.

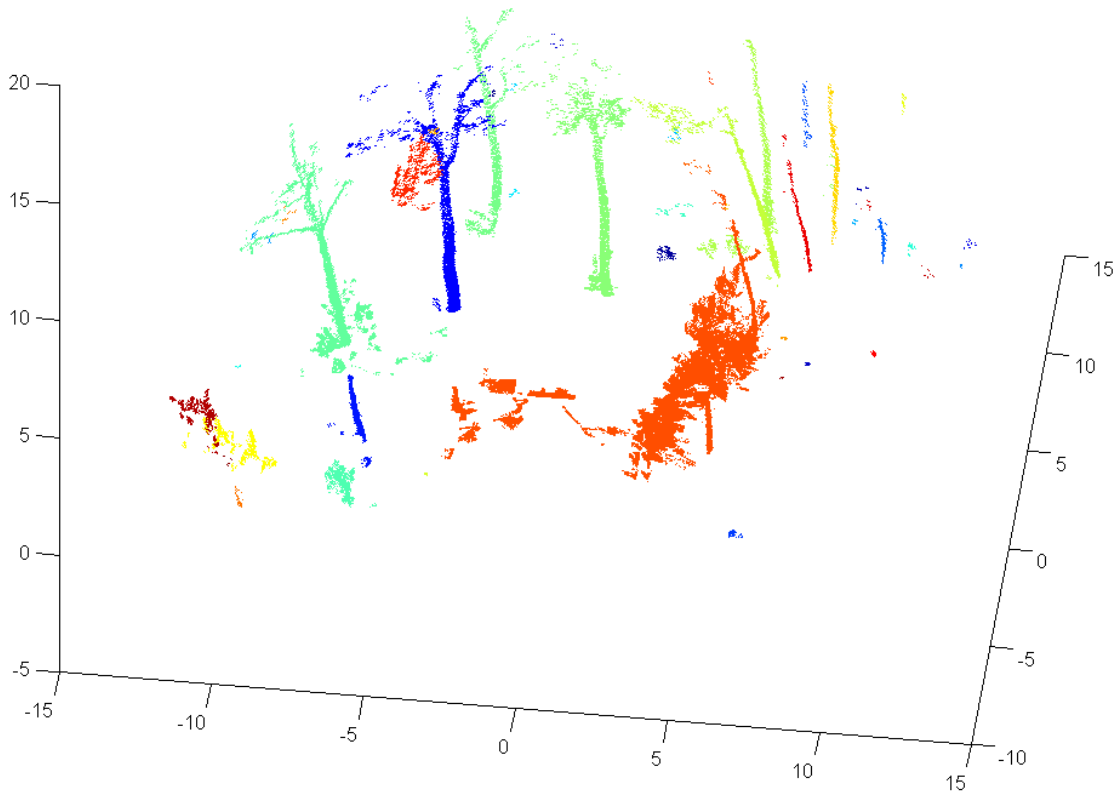


Fig. 4.4 Scene segmented into objects with no adjacent voxels between them. Different colors indicate different objects. Ground points are not shown.

### 4.2.3 Evaluation

#### Rules application evaluation

This section analyzes the performance of the proposed segmentation algorithms using different order or combination of the refining rules. The evaluation method uses three scenes:

- *Urban*. This dataset is described in chapter 3 and it allows the comparison with other author's methodology [26]. It contains 41 large objects including buildings, trees, cars, bushes, fences and light poles.
- *Dense*. This dataset is described in chapter 3 where is called *Natural\_2*. It represents a dense forest with many trees and bushes over the ground.
- *Slope*. This dataset shows a natural environment in Malaga captured by a low-cost laser range finder called Unolaser [63] (see Fig. 4.8). It contains a steep slope and just a tree and a bush.

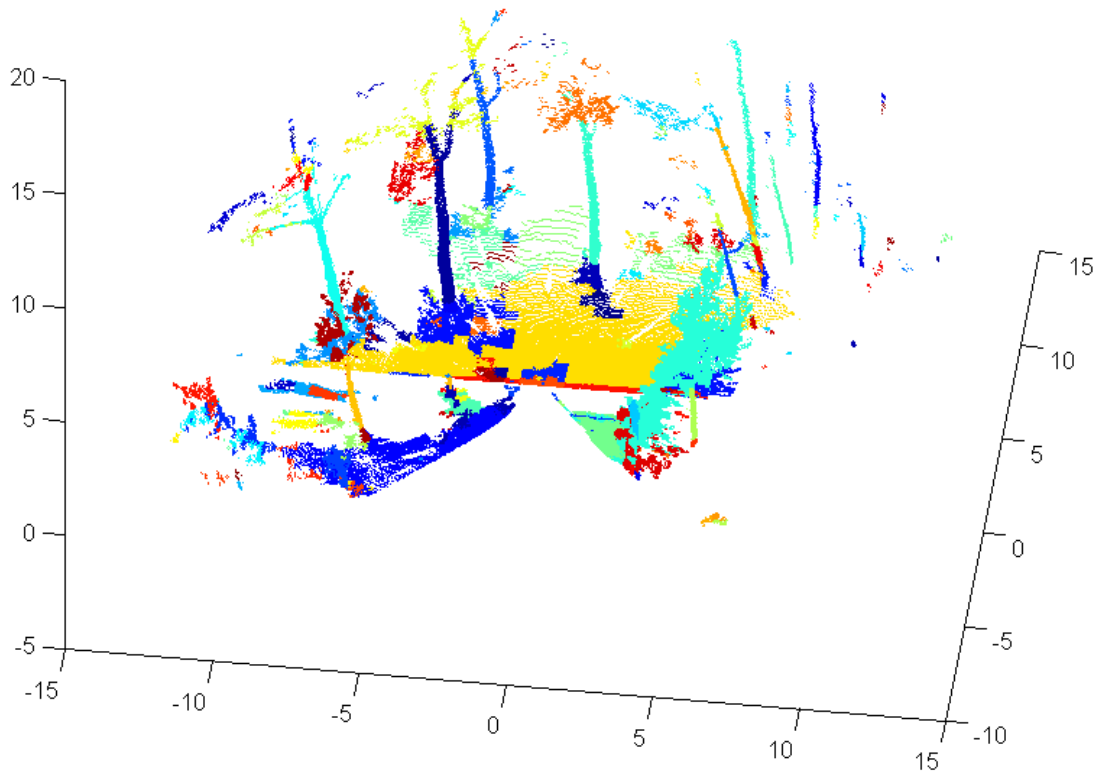


Fig. 4.5 Scene with objects segmented into regions with the same geometric class: planar, linear, scatter. Different colors indicate different segments.

Noisy points from sparse areas will not be taken into account in the evaluation since they cannot be manually identified as an object. Initially, a voxel size  $E$  is 0.5 meters, however owing to the fact that  $E$  affects the accuracy in the segmentation process, this parameter is tested in Section 4.2.3 .

Through four metrics ( $\mathbb{G}$ ,  $\mathbb{F}$ ,  $\mathbb{O}$ ,  $\mathbb{N}$ ) the evaluation quantifies the agreement between the manual segmentation and the segmentation resulting from the proposed algorithm. That is, each hand-labeled object or ground  $U$  is compared with the same points in the scene segmented with the proposed algorithm. The evaluation method considers two metrics for the ground segmentation (i.e.  $\mathbb{G}$ ,  $\mathbb{F}$ ) and other two metrics for the object segmentation (i.e.  $\mathbb{O}$ ,  $\mathbb{N}$ ). This set of metrics aims to balance the effects of both oversegmentation and undersegmentation. Oversegmentation happens when a hand-labeled object  $U$  wrongly corresponds to more than one object,  $\{U', V', W', \dots\}$ , in the scene segmented with the algorithm (see Fig. 4.6a). That is,  $U' \cup V' \cup W' = U$  where  $U' > V' > W'$  (in number of contained



Table 4.1 Segmentation Results with Different Ground Segmentation Rule Sequences

Rule Sequence	Slope Scene					Dense Scene					Urban Scene					Mean[%]
	G[%]	F[%]	O[%]	N[%]	T[s]	G[%]	F[%]	O[%]	N[%]	T[s]	G[%]	F[%]	O[%]	N[%]	T[s]	
<i>ABC</i>	50,8	0,0	61,5	0,6	4,0	99,3	4,8	61,2	36,3	6,1	100,0	0,1	94,7	94,7	85,3	43,9
<i>ABCD</i>	66,0	0,0	61,5	0,6	3,5	99,9	25,0	50,6	32,2	4,3	100,0	2,4	93,4	93,4	113,9	42,1
<i>ABCD'</i>	69,8	1,4	32,4	0,6	2,7	99,9	62,6	45,9	36,4	4,5	100,0	3,6	92,7	92,7	91,6	43,2
<i>ABDC</i>	65,3	0,0	61,5	0,6	2,3	99,7	18,6	61,0	54,4	3,0	100,0	2,2	93,4	93,4	97,2	49,5
<i>ABD'</i>	69,3	1,4	32,4	0,6	2,7	99,6	51,5	50,3	40,8	1,5	99,6	3,3	92,8	92,8	100,6	44,7
<i>ABD'C</i>	69,8	1,7	26,5	0,6	2,5	99,7	55,6	47,9	38,4	5,8	100,0	3,3	92,8	92,8	100,3	43,9
<i>CAB</i>	50,4	0,0	61,5	0,6	2,6	99,3	4,7	61,0	36,3	6,1	99,9	0,1	94,7	94,7	102,1	43,9
<i>CDAB</i>	69,8	0,0	61,5	0,6	4,3	99,9	28,3	48,6	30,4	4,2	99,9	2,4	93,5	93,5	112,3	41,5
<i>CD'AB</i>	73,0	0,0	61,5	0,6	2,7	99,9	64,8	44,8	35,3	6,8	99,9	3,4	93,0	93,0	113,6	43,0
<i>DCAB</i>	65,4	0,0	61,5	0,6	4,7	99,7	21,3	59,1	52,4	6,0	99,9	2,1	93,5	93,5	101,5	48,9
<i>D'AB</i>	72,6	0,0	61,5	0,6	3,5	99,6	54,4	48,4	38,9	3,8	99,6	3,2	93,0	93,0	98,5	44,2
<i>D'CAB</i>	72,9	0,0	61,5	0,6	2,4	99,7	56,7	47,0	37,5	3,6	99,9	3,2	93,0	93,0	84,8	43,7
$\emptyset$	54,6	0,0	61,5	0,6	2,2	98,4	5,5	55,7	30,8	4,5	88,5	2,1	80,4	70,5	109,1	34,0
<i>D'</i>	98,4	2,8	38,2	38,2	1,7	99,6	75,5	36,1	26,8	2,8	99,6	6,3	90,0	90,0	125,5	51,7
<i>D'B</i>	98,4	2,8	38,2	38,2	3,5	99,6	73,0	36,9	27,1	4,1	99,6	3,8	93,1	93,1	112,0	52,8
[26]	85,5	4,6	61,5	0,6	4,3	48,9	4,3	57,3	32,1	4,8	100,0	0,6	94,3	94,3	117,3	42,3

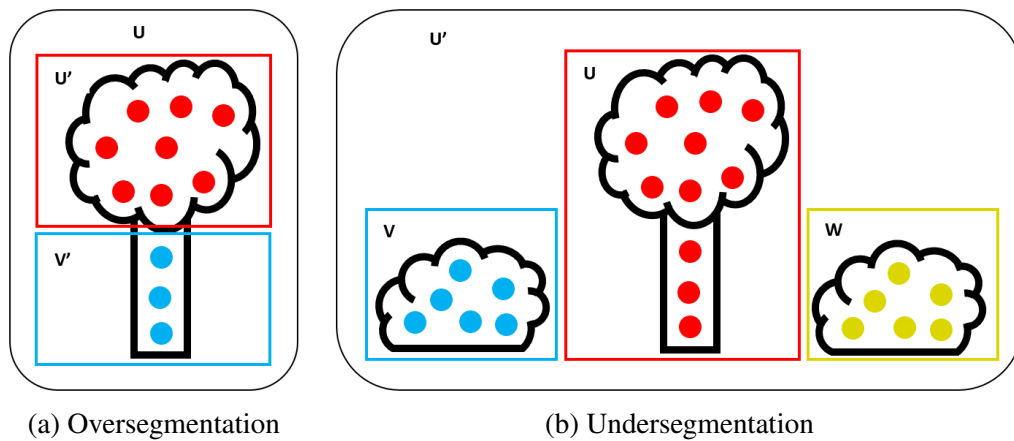


Fig. 4.6 Errors in segmentation.  $U$ ,  $V$ ,  $W$  are hand-labeled objects while  $U'$ ,  $V'$  are objects segmented with the proposed segmentation algorithm. (a) Represents an oversegmented scene and (b) the same scene with undersegmentation.

points). Undersegmentation occurs when several hand-labeled objects,  $\{U, V, W, \dots\}$ , have been segmented in only one object  $U'$  (see Fig. 4.6b). That is,  $U \cup V \cup W = U'$ .

The first metric  $\textcircled{O}$  is computed by iterating for all the hand-labeled objects, sorted in descending order of size. The size of an object  $U$  is the number of its inner points and it is denoted by  $s(U)$ . The agreement score for  $U$  is computed only with respect  $U'$  as  $s(U' \cap U) * 100 / s(U)$ , in percentage, where  $U'$  is the largest object (in number of points) in the set of points in the scene segmented with the algorithm which corresponds to  $U$ . So the largest object  $U'$  is considered the match,  $U \sim U'$ , and all other partitions derived from oversegmentation,  $V', W'$ , are considered errors because  $\nexists V, W : V \sim V', W \sim W'$ . Furthermore, to quantify the error from undersegmentation, once a set of points in  $U'$  has

been matched to any  $U$ , then the whole object  $U'$  cannot be considered in any subsequent match iterations for the remaining hand-labeled objects. That is, if  $U' \sim U \implies U' \approx V$ . This way, in oversegmentation of the object  $U$  the score will be  $\mathbb{O}(U) \in (0, 100)$  while in undersegmentation:  $\mathbb{O}(U) \in (0, 100], \mathbb{O}(V) = 0$ . This metric was proposed in [26] where the author did not consider the ground as a special object, and the undersegmentation is not quantified enough.

Moreover, this chapter proposes a novel metric  $\mathbb{N}$  which considers the undersegmentation as a bigger penalty. This metric is similar to  $\mathbb{O}$  but 0% score is assigned if  $U'$  has more points outside of  $U$  than inside it (see Fig. 4.6b). Then, for  $U \cup V \cup W = U' : U < V \cap W \implies \mathbb{N}(U) = \mathbb{N}(V) = \mathbb{N}(W) = 0$ . Hence,  $\mathbb{N} \leq \mathbb{O}$ .

The metric  $\mathbb{G}$  computes the agreement of the ground segmentation by applying the same score equation as  $\mathbb{O}$  to the hand-segmented ground.

Furthermore, the false positive points could be measured through the metric  $\mathbb{F}$ . That is, the number of points outside the hand-labeled object which have been wrongly segmented in its matched object, with respect to the total number of points of the hand-labeled object, i.e.  $s(U' \cdot U) * 100/s(U)$ . This score measures the undersegmentation, in this case the score will be a positive number, while in oversegmentation the score is 0. This section applies  $\mathbb{F}$  only to the ground segmentation.

Table 4.1 shows the evaluation results with 15 representative combinations of the refining rules of the ground segmentation. The first column indicates the rule sequence; rule  $\emptyset$  corresponds to the rough ground estimation and no refining rule applied. Then the four metrics are applied to each of the three evaluated datasets and the execution time of the ground and object segmentation, in seconds. The last column represents the mean over the three datasets evaluated with the novel  $\mathbb{N}$  metric. Besides, the table offers metrics for the method with the best result proposed in [26] called Cluster-All. For this method, the parameter neighborhood magnitude is defined as 1, which means two voxels are considered neighbors only if they are touching, adjacent, as in the proposed algorithm. In addition, the variance and height thresholds for the ground required by the cluster-all method were empirically extracted from the hand-labeled Urban scene. The algorithms have been coded in Matlab and executed on a computer with a i7 processor with a clock frequency of 2.4 GHz and 8GB in RAM.

In general, the Urban scene presents the best result regarding ground and object segmentation in all rule combinations, but the computational times are large because of the large number of points. This is a structured environment where the ground is continuous and planar and the objects are clearly separated. The Dense scene presents very good results in the ground extraction, but between 30 and 50% of point score in the object segmentation. This

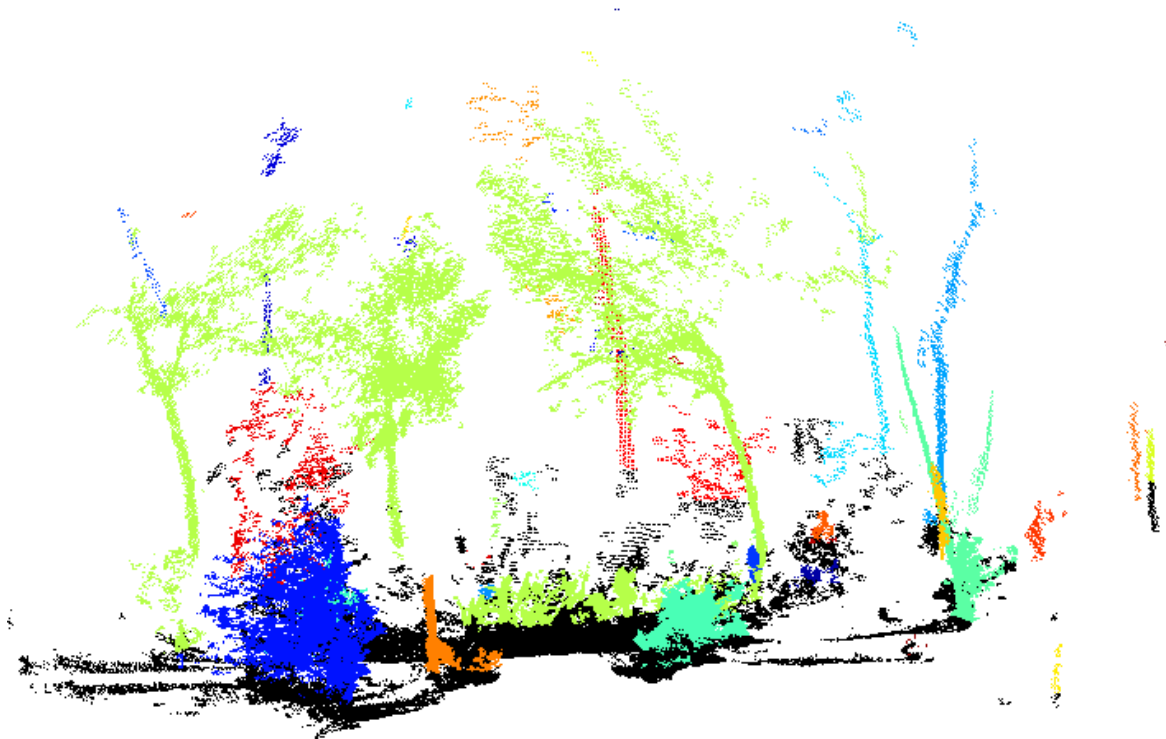


Fig. 4.7 Dense scene segmented with the method *ABDC*. Black points are segmented as ground, and different colors indicate different segmented objects. The largest three trees are joined by their crowns and some bush.

is because all the vegetation, bushes and tree crowns, are merged with other objects in the scene (see Fig. 4.7). In fact, it may not present a problem for future navigability assessment because the second segmentation in regions shows the parts of the object which are solid obstacles (linear or tubular class) or traversable such as bushes on the ground (scatter class). Finally, the most difficult scene is the Slope scene where the ground segmentation has scores between 50 and 73% in most of the methods, except the last three, and this produces poor object segmentation. In addition, this scene shows the lack of accuracy in both metrics for object segmentation. Fig. 4.8 shows the Slope scene segmented with the rule sequence *ABCD*. The main object is the tree which has been wrongly segmented, from an area of the ground which is also badly segmented. The  $\odot$  metric provides a score of 61% and the second metric 0.6% which is more accurate for representing this undersegmentation case.

To sum up, there is no method with the best score in all the scenes. However, the mean of the proposed metric  $\mathbb{N}$  in the three scenes shows that the optimal combination of refining rules is *D'B*. These correspond to the sequence of adding voxels in a slope (iterative version) and removing voxels in isolated areas with vertical discontinuity. When the removing rules (*A* or *B*) are applied before the adding rules (*C*, *D* or *D'*), some voxels required for the

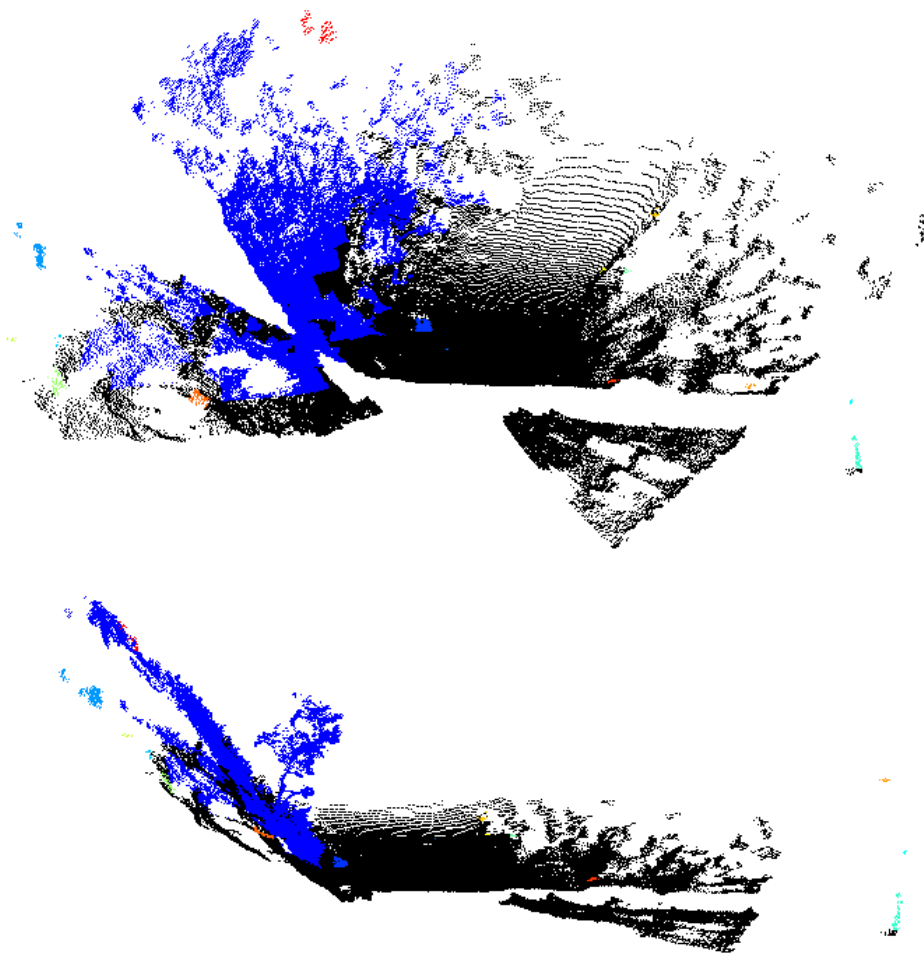


Fig. 4.8 Slope scene wrongly segmented. The bottom image shows the same scene from the side. Black points are segmented as ground, and different colors indicate different segmented objects.

interpolation in a slope ground are removed. However, in scenes where the ground is almost flat and continuous, removing rules applied in the first place provides better computation time, because the subsequent rules will process less data, and no accuracy penalty is shown. The segmentation algorithm proposed in this chapter obtains a much better result than other in the literature especially in natural environments with big slopes.

### Voxel Size Evaluation

This section evaluates the proposed segmentation algorithm, with its optimal method  $D/B$ , with different voxel sizes.

A too small size of voxel produces sparser data, the number of empty voxels which connect surfaces from the same object increases, then objects become over-segmentated.

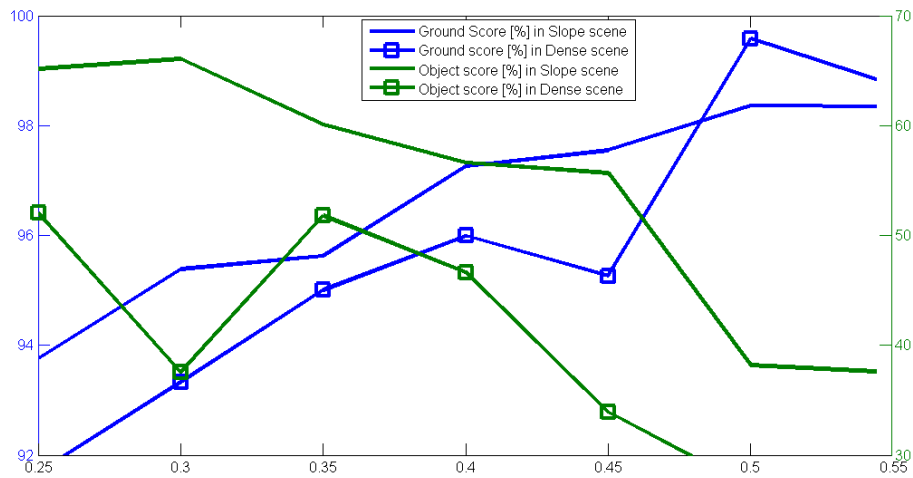


Fig. 4.9 Ground and Object point scores, in percent, for the data set Slope and Dense with different voxel sizes. The ground scores are in the blue scale, in the left vertical axis, and the object scores are in the green scale, in the right vertical axis. X axis is the size of the voxel tested, in  $\text{m}^3$

On the contrary, too big voxels can create adjacency in voxels which really are in different objects. This could be an advantage in sparse point clouds where two points are separated but their voxels can offer a direct adjacency, hence the segmentation process will merge these points in the same object. In dense point clouds, the points are closer and the optimal voxel size will be the one which is as big as possible to further simplify the point cloud but at the same time small enough to keep the meaningful features of the scene, as shapes.

Fig. 4.9 shows the ground and object segmentation scores,  $\mathbb{G}$ ,  $\mathbb{N}$ , for Slope and Dense scenes. When the voxel size increases, the ground score also increases however the object score decreases. In conclusion, more accurate segmentation results are obtained with lower voxel sizes. Furthermore, the performance of the algorithm is analyzed in terms of time and object score per unit of time. Fig. 4.10 presents the decrease of the execution time as the voxel size increases, as it would be assumed that the point cloud is becoming simpler in a smaller number of voxels. However, since the available time is limited in many applications, it is useful to analyze the performance in terms of the segmentation score per each unit of the executed time. The rate of score per time is based on the result of the Point Score in the object segmentation with the novel metric  $\mathbb{N}$  in percent, per each unit of the execution time in seconds. The figure shows that the optimal performance is obtained with a voxel size between  $0.45$  and  $0.5 \text{ m}^3$ .

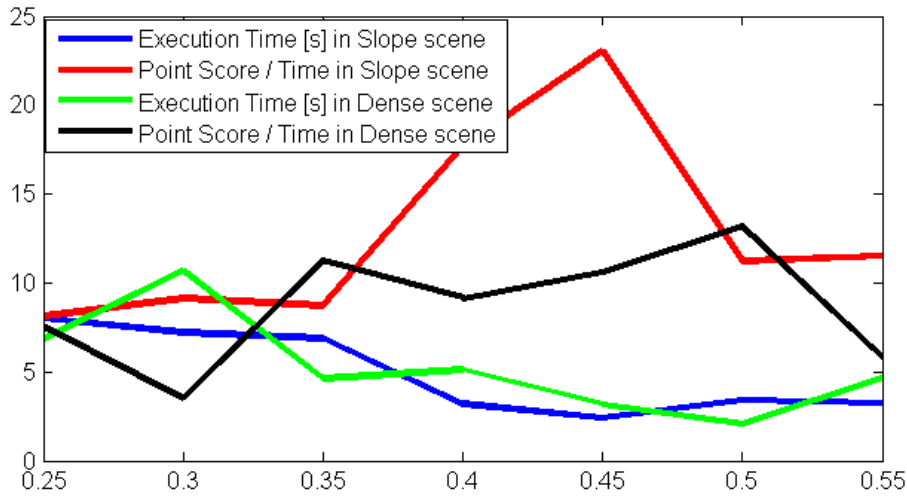


Fig. 4.10 Execution times in seconds and rates of segmentation scores per unit of time with different voxel sizes. Blue and green lines are the execution times in Slope and Dense scene respectively, and red and black lines are the score per time rate for these scenes, respectively. X axis is the size of the voxel used, in m<sup>3</sup>

### 4.3 Navigability assessment

A 2D occupancy grid can be obtained based on the extracted attributes, the geometric class and the segmentation in ground or object, and two parameters  $\eta_h$  and  $\eta_l$ , which are height thresholds for the obstacles and depend on the vehicle features.

The goal is reducing the dimensions of the 3D grid composed by voxels by projecting it into a new 2D  $XY$  grid composed by cells. The result is a  $XY$  plane divided into equal size square cells which are classified in accessible and non-accessible area. The plane  $XY$  correspond to the 3D grid plant so the size of the cells edge is the same as the voxels edge. The value for each cell is computed based on the features of the voxels over the current cell. A flow diagram is depicted in Fig. 4.11.

When no voxel exists on the vertical of a  $XY$  cell, then occupancy cell will be empty because there is no point on it, or unknown data. If there is no voxel upper than the ground voxel on the current  $XY$  cell, then the cell is accessible, assuming all ground voxels are traversable for the vehicle. Otherwise, the height of the voxels which are non ground over the current cell,  $VO$  voxels, determines the output value for the occupancy cell. The parameters  $\eta_h$  and  $\eta_l$  correspond to the high and low threshold of the collision zone for the vehicle: objects higher than the collision zone are not obstacles in the path, objects in that zone are always obstacle, and objects under that zone can be depending on other features. These parameters assume the ground's height is zero. However, since the ground has slopes or

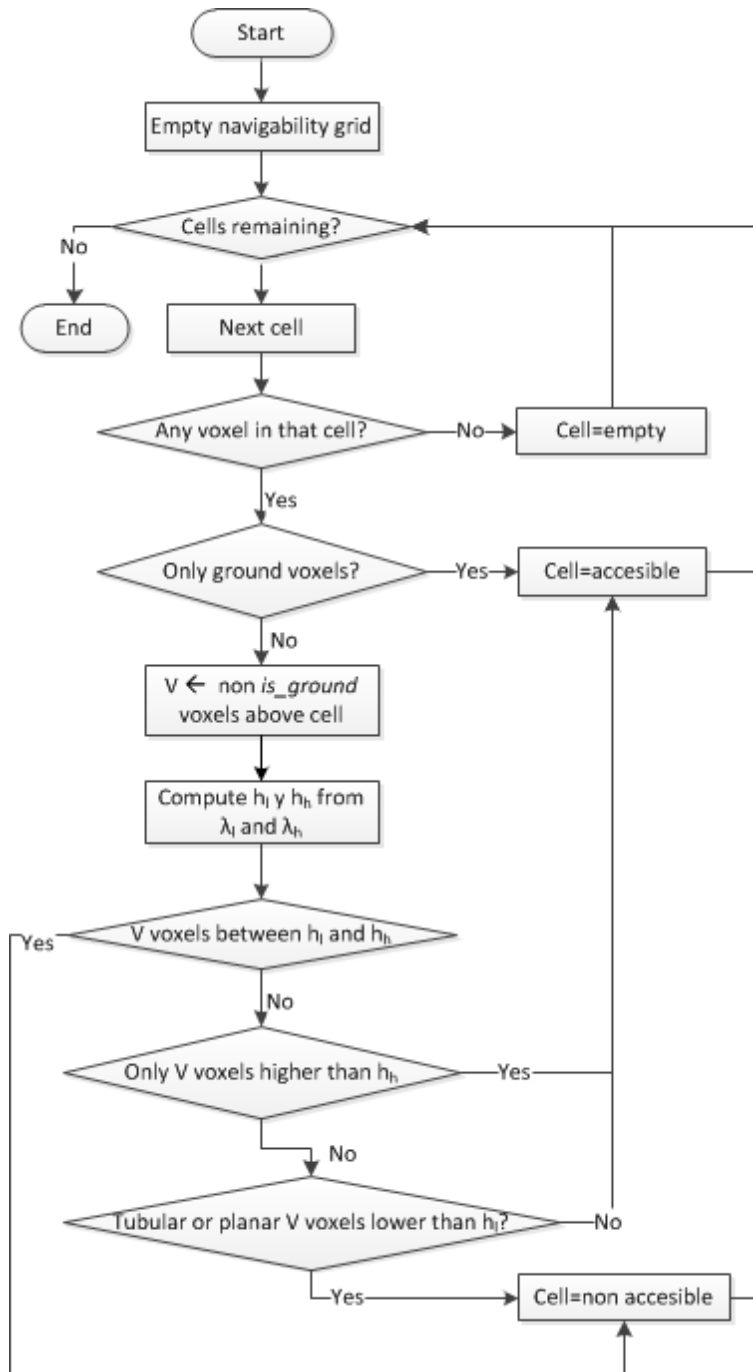


Fig. 4.11 Flow diagram of the occupancy grid generation



roughness,  $\eta_h$  and  $\eta_l$  are transformed into  $h_h$  and  $h_l$  which are adapted to the height of the ground at each cell. A cell with voxels upper than the high threshold  $h_h$  over its ground voxel means the obstacle is too high for a collision with the vehicle, hence it is an accessible cell. On the contrary, if voxels between the high and low threshold  $h_h$  and  $h_l$  exist, then an obstacle exists in the vehicle collision zone, therefore the cell is declared as non-accessible. In the case of voxels lower than  $h_l$ , the objects has a height accessible for the vehicle depending on its strength. Voxels classified as tubular or planar surface, are considered solid objects hence non-accessible obstacle. However, scatter voxels usually referred to vegetation are accessible when they are as short as the  $h_l$  defines. As a result, a binary occupancy grid is generated which can be included in a robot navigation system for planning trajectories.

### 4.3.1 Evaluation

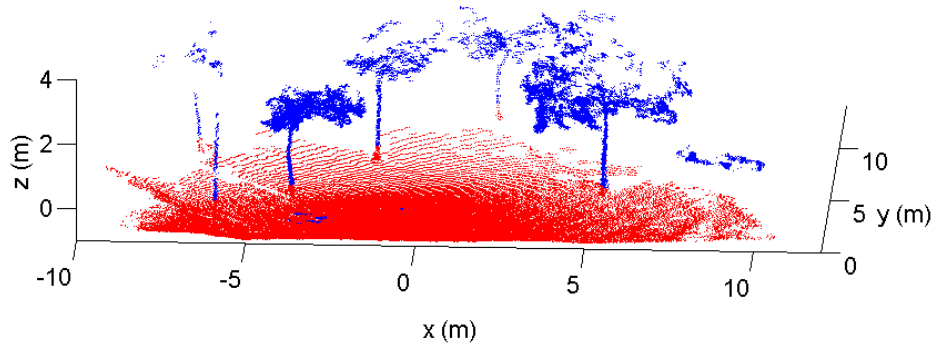
This section presents the results of the proposed occupancy grid generation tested in several point clouds captured with a 3D laser rangefinder in natural environments.

The first step of the proposed method involves transforming the raw data from the laser rangefinder to a point cloud such as list of Cartesian 3D points. This process is different depending on the 3D device used. In this analysis, the low-cost laser rangefinder UnoLaser 30M135Y has been used [63]. It is a stop-and-go laser system [30] based on pitching a commercial 2D rangefinder, Hokuyo UTM-30LX. It has a maximum scanning range of 30 meters, a field of view 270 in horizontal and 131° in vertical. A spherical scan is obtained where each point in the cloud is defined by two angles  $\omega$  and  $\alpha$  around Z and X axes, respectively, and a distance  $R$ . The Cartesian coordinates  $[x, y, z]^T$  are computed based on the spherical coordinates  $[R, \omega, \alpha]$ .

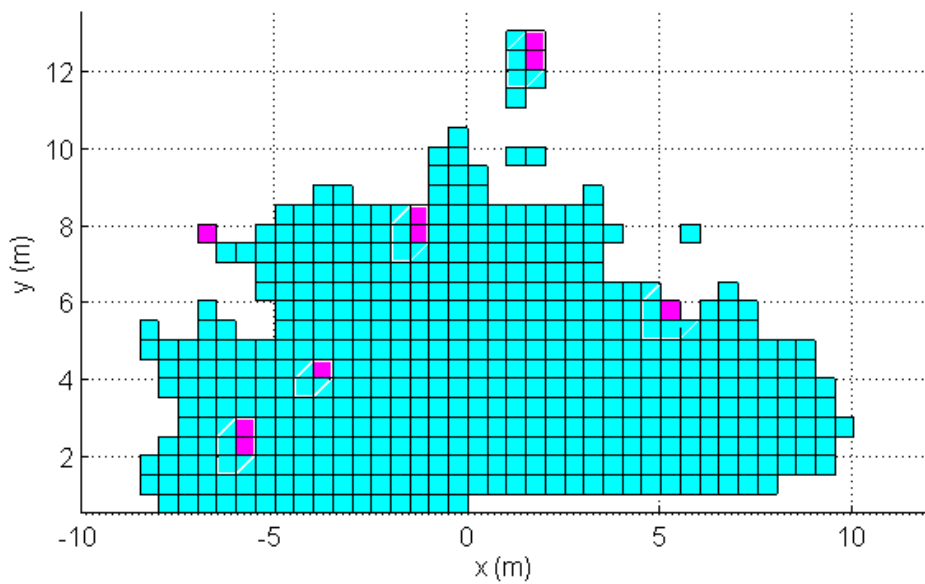
Point clouds used in experiments correspond to environments captured during a realistic emergency drill in natural environments in Marbella, Spain [14]. In the voxel map generation, the parameter voxel edge size  $E$  has been defined as 0.5 meters achieving a compromise between computational efficiency, memory management and scene reconstruction accuracy. The neural network has been previously trained with several hand-labeled scenes which are different to the scenes used in the method evaluation. Same way the density threshold  $\rho$  is 10, which means voxels are processed only if contains at least 10 points. The vehicle parameters for the occupancy grid generation are  $\eta_h = 1.5$  meters and  $\eta_l = 0.5$  meters.

As expected, the resulting occupancy grids present non-accessible cells as obstacles exist over the ground. Fig. 4.12 shows the generated occupancy grid based on a natural scene with trees. The tree crowns are too high for the vehicles parameters, for that reason these cells are accessible. On the contrary the tree trunks are obstacles for the navigation are labelled as non-accessible. Other scene has been processed with a slope, see Fig. 4.13. Even with





(a) Point cloud from a trees scene where ground is marked in red and left objects in blue



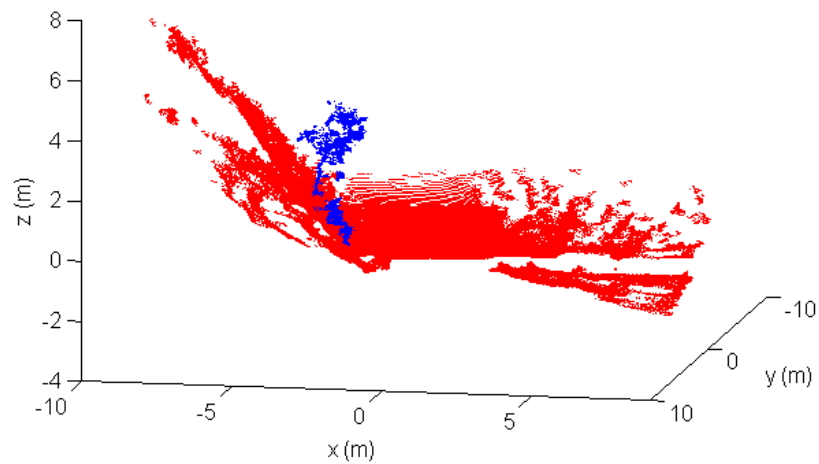
(b) Resulting occupancy grid where accessible area is colored in blue and obstacles in pink.

Fig. 4.12 Occupancy grid resulting in a trees scene

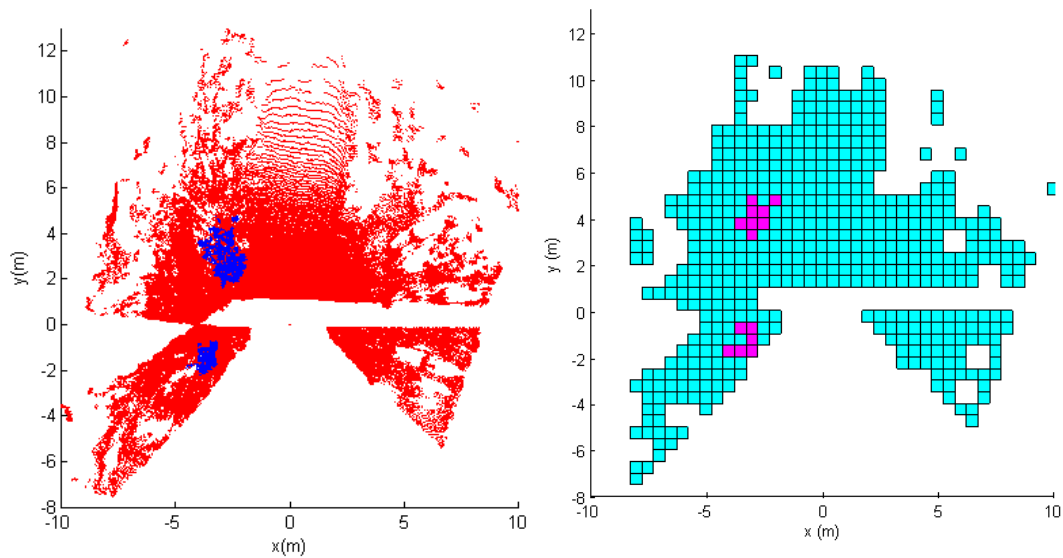
the high slope of the ground these cells are considered accessible for the vehicle navigation. However, the tree and the bush are labelled as obstacles.

## 4.4 Summary

This thesis has proposed a new method for 3D segmentation and 2D occupancy grid generation where cells are labelled as accessible or non-accessible, both specially designed for unstructured environments where the ground is not flat or continuous. The methods are based in GFVM described in chapter 3. The ground is segmented from the scene where big slopes, non-flat areas, and isolated areas which may appear in unstructured environments have been considered. To identify these ground voxels, the ground segmentation algorithm consists



(a) Point cloud from a slope scene where ground is marked in red and left objects in blue



(b) Plant view from the same point cloud.

(c) Resulting occupancy grid where accessible area is colored in blue and obstacles in pink.

Fig. 4.13 Occupancy grid resulting in a slope scene

of a sequence of refining rules. An appropriate sequence for the rules has been determined from experimental analysis. Furthermore, the influence of voxel size has also been analyzed. Besides, ground segmentation is used to extend the voxel map representation by adding a new attribute to the voxels, apart from their geometric class.

Once the ground is segmented from the scene, the object segmentation is performed in two steps: *i*) adjacent voxels define segmented objects, and *ii*) each object is splitted into regions with adjacent voxels and the same geometric class. With this double-segmentation method the traversable ground can be extracted to facilitate the path planning. In addition, several geometric and statistical features can be extracted from both objects and regions in order to classify objects automatically and in the recognition of the scene. Hence, the proposed object segmentation and characterization method can be useful for further scene understanding processing problems, such as semantic mapping and scene recognition.

Generation of the occupancy grid is based on the location and the geometric features of the objects over the ground. Those voxels classified as tubular or planar surfaces, except the ground, are assumed as solid objects or obstacles, hence they may be semantically labeled as non-accessible areas. Furthermore, scatter voxels can be an obstacle for a vehicle depending on its height. Thus, the proposed solution aims to develop a low computational cost method by reducing the amount of data to process in comparison with point-based techniques. Furthermore, geometric features are identified regardless of their orientation, which extends applicability to unstructured environments. As a result, an occupancy grid is generated which can be included in a robot navigation system for planning trajectories.

This chapter has presented a novel approach to obstacle identification for autonomous ground vehicles that is applicable to both structured (e.g. indoor, road) and unstructured (e.g. off-road, grassy terrain) environments. The feasibility of this method has been evaluated with 3D laser scans in urban and natural environments using data acquired from range sensors widely studied. The proposed algorithm presents better results in comparison with other methods from the literature in natural environments with rough ground. In addition, an experiment conclude which size of voxel provides the best result. Apart from a metric commonly used in the literature, a novel metric has been used is to quantify the accuracy in the segmentation trough a comparison between each segmented point cloud and a manually labeled ground truth.

The scientific production of this chapter is listed below:

- [84] ICIT' 15, IEEE International Conference on Industrial Technology. 3D Segmentation Method for Natural Environments based on a Geometric-Featured Voxel Map. 2015.

## **64** Segmentation and navigability assessment based on geometrically featured voxel maps

- [87] MED'15, 23rd Mediterranean Conference on Control and Automation. Occupancy grids generation based on Geometric-Featured Voxel maps. 2015.

# Chapter 5

## Analysis and construction of a multi-beam lidar with an additional degree of freedom

### 5.1 Introduction

This chapter addresses RMBL sensors, a type of customized 3D rangefinders built by adding a rotation mechanism to a commercial MBL. Recent published examples using VLP-16 (Puck) [71][44], the most affordable and lightest MBL by Velodyne, indicate that the RMBL configuration has the potential to become a common solution to get low-cost, rapid, and full-3D high resolution scans, as has happened with customized RSML during the last fifteen years (as shown in Table 2.1).

In particular, novel contributions include a comparison, both qualitative and quantitative, of general scan patterns produced by an ideal VLP-16-based RMBL with other 3D scan alternatives (i.e., SMBL and MBL), the design and implementation of a new RMBL consisting of a portable tilting platform for Velodyne VLP-16 scanners, and a discussion of actual example scans obtained with the new device.

The rest of the chapter is organized as follows. Section 5.2 defines an ideal RMBL based on Velodyne MBL specifications. Section 5.3 presents a simulation-based analysis methodology. Section 5.4 analyzes RMBL scan patterns and offers comparisons with alternative 3D configurations. Section 5.5 describes a new VLP-16-based tilting platform, which is used to scan three representative environments that are discussed in Section 5.6. Finally, Section 5.7 offers the summary.

## 5.2 Rotation of a multi-beam lidar sensor

This section reviews the major characteristics of two MBL sensors by Velodyne and presents the computation of Cartesian point clouds for an RMBL that considers a rotation axis parallel to one of the MBL frame axes.

### 5.2.1 Commercial multi-beam lidars

As opposed to single-beam 2D sensors, in multi-beam rangefinders the rotating mirror is replaced by a spinning structure that holds a number of independent laser transceivers to scan different elevation angles. The Velodyne VLP-16 and HDL-32 rangefinders, shown in Fig. 5.1, are representative examples of the most affordable end of commercial multi-beam sensors [109].

Table 5.1 Manufacturer specifications for the VLP-16 and HDL-32 sensors [10].

	VLP-16	HDL-32
Laser/detector pairs	16	32
Range	1m to 100m	1m to 70m
Accuracy	$\pm 3\text{cm}$	$\pm 2\text{cm}$
Data	Distance / Calibrated reflectivities	Distance / Calibrated reflectivities
Data Rate	300,000 points/s	700,000 points/s
Vertical FOV	$30^\circ : [-15^\circ, +15^\circ]$	$41.3^\circ : [-30.67^\circ, +10.67^\circ]$
Vertical Resolution	$2.0^\circ$	$1.33^\circ$
Horizontal FOV	$360^\circ$	$360^\circ$
Horizontal Resolution	$0.1^\circ$ to $0.4^\circ$ (programmable)	$0.08^\circ$ to $0.35^\circ$ (programmable)
Size	103mm x 72mm	85.3mm x 149.9mm
Weight	0.83Kg	1.3Kg



(a) Velodyne HDL-32E (b) Velodyne VLP-16

Fig. 5.1 Commercial multi-beam lidars [10].

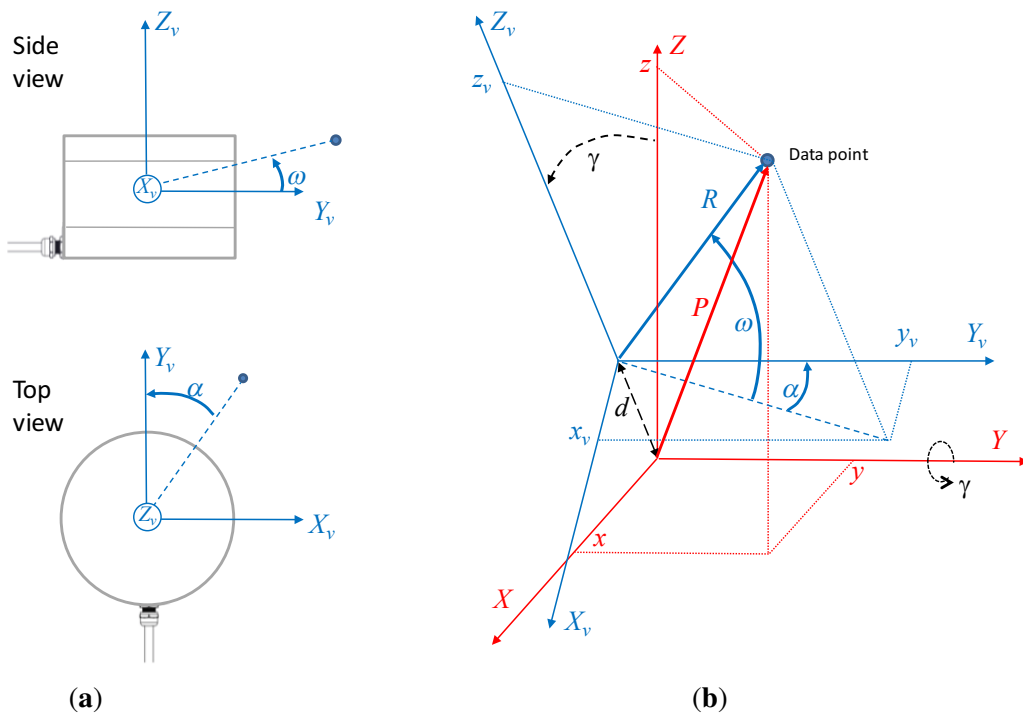


Fig. 5.2 Reference frames and data point coordinates: a) Velodyne VLP-16 sensor, b) RMBL based on a VLP-16. The VLP-16 local frame is represented in blue; the RMBL local frame is represented in red.

The major specifications of the VLP-16 and HDL-32 MBLs are summarized in Table 5.1. From an operational standpoint, the major differences between them lie in the number of laser transceivers and the vertical FOV: The VLP-16 scanner has 16 individual laser/detectors arranged in a  $30^\circ$  FOV, which yields a vertical resolution of  $2.0^\circ$ , whereas the HDL-32 has 32 transducers within a FOV of  $41.3^\circ$  with a vertical resolution of  $1.33^\circ$ . In contrast with the VLP-16, whose FOV is symmetrical with respect to the horizontal plane, the FOV of the HDL-32 has a downward shift. Furthermore, the increased vertical resolution of the HDL-32 has a significant impact in the cost of the sensor, which is substantially more expensive than the VLP-16.

### 5.2.2 Multi-beam lidar with an additional rotation (RMBL)

Without loss of generality, the VLP-16 will be considered in this Thesis for the addition of a rotation mechanism. This sensor is especially suitable to build an RMBL on account of its more accessible cost, lighter weight, symmetric FOV, and compact size. Nevertheless, the following definitions could be extended to any MBL.

The local frame  $X_v Y_v Z_v$  of the VLP-16 is shown in Fig. 5.2(a). This frame has its origin in the optical center, with the  $Y_v$  axis in the forward direction and  $Z_v$  pointing upwards. The VLP-16 scans points in spherical coordinates  $(R, \omega, \alpha)$ . With this information, Cartesian coordinates  $(x_v, y_v, z_v)$  can be obtained for each measured point:

$$x_v = R \cos(\omega) \sin(\alpha), \quad (5.1)$$

$$y_v = R \cos(\omega) \cos(\alpha), \quad (5.2)$$

$$z_v = R \sin(\omega). \quad (5.3)$$

The local frame  $XYZ$  of the RMBL resulting from the addition of a rotating mechanism (i.e., spinning or tilting) to the VLP-16 is illustrated in Fig. 5.2(b). When the rotation angle  $\gamma$  is null,  $Z_v$  is aligned with  $Z$ , and  $X_v$  and  $Y_v$  are parallel to  $X$  and  $Y$ , respectively. Let us consider that the rotation axis is parallel to one of the VLP-16 axes; in this case, the rotation axis is  $Y$ . It should be noted that rotation about the  $Y$  and  $X$  axes would be similar, as the VLP-16 has a  $360^\circ$  horizontal FOV, whereas rotation about  $Z$  would be pointless as this is redundant with the spinning motion of the multi-beam transceivers. Furthermore, in practice, the rotation axis should be at some small distance  $d$  below the VLP-16 for the sake of compactness and shadow avoidance.

Then, Cartesian coordinates  $(x, y, z)$  of data points in the RMBL frame can be computed as:

$$x = R(\cos(\omega) \sin(\alpha) \cos(\gamma) + \sin(\omega) \sin(\gamma)) + d \sin(\gamma), \quad (5.4)$$

$$y = R \cos(\omega) \cos(\alpha), \quad (5.5)$$

$$z = R(\sin(\omega) \cos(\gamma) - \cos(\omega) \sin(\alpha) \sin(\gamma)) + d \cos(\gamma). \quad (5.6)$$

### 5.3 Analysis methodology

This section proposes a simulation-based methodology to analyze, both qualitatively and quantitatively, the spatial distribution of laser beams in 3D lidars. First, the simulation of sensor points on a hollow sphere is defined. Qualitative analysis is done from a visualization of point patterns projected on the sphere and also on orthogonal planes. For a quantitative analysis, the distribution of sampling density is considered.



### 5.3.1 Numerical simulation

The proposed methodology considers the set of points computed by simulating a scan from a sensor that is placed at the center of a virtual hollow sphere [114][96]. This structure allows analyzing the homogeneity and the beam density in all directions around the sensor. By using a sphere, the analysis depends exclusively on sensor characteristics and is independent of the orientation and distance of the target surface (e.g., as in planar targets). Moreover, the azimuth and elevation of the points in the sphere are independent of the sphere radius.

To analyze the general patterns produced by different scan configurations, Section 5.4 will consider ideal sensors that are independent of particular device considerations. In order to produce a complete sphere, an ideal RMBL sensor based on a VLP-16 lidar can be simulated by considering a constant angular velocity of the tilt motion from  $\gamma = 0^\circ$  to  $\gamma = 180^\circ$ , which produces a complete sphere. Different angular velocities correspond to different scan resolutions for the additional DOF. For the sake of simplicity, it will also be assumed that the sphere radius is large enough to make the deviation between the center of the sphere (i.e., the RMBL's origin) and its optical center negligible (i.e.,  $d \approx 0$ ); thus, all ranges  $R$  coincide with the sphere radius and Equations (5.1)-(5.3) can be applied. Moreover, as points are intended to represent beam directions, no noise is considered in the simulations. Besides, for generalization, no shadows or other FOV limitations due to a particular mechanism are considered.

For a qualitative analysis, it is also interesting to consider how the angular distribution of points on the sphere would be translated onto planar surfaces in a synthetic environment. In the proposed analysis, the sensor is placed at a height of 1.5 m over a ground plane that is parallel to the local  $XY$  plane and at 10 m from planes (representing walls) that are parallel to  $YZ$  and  $XZ$  planes. This synthetic configuration is preferred rather than placing the sensor at the center of a  $10 \text{ m} \times 10 \text{ m}$  cube [1] because it is more representative of ground-based lidar applications.

### 5.3.2 Sampling density

Sampling density for different scanning angles [114][1] can be represented as a 2D histogram on the sphere. With this aim, the sphere surface is partitioned with a triangular mesh obtained by recursive icosahedron sphere tessellation [7]. In this work, the sphere surface has been partitioned into 5120 bins.

## 5.4 Analysis of ideal RMBL scanning patterns and comparison with other 3D configurations

The methodology defined in Section 5.3 has been applied to an ideal full-sphere VLP-16-based RMBL, which is compared with alternative 3D lidar configurations.

### 5.4.1 Qualitative analysis

A visualization of sphere points given by the 3D scanners is offered in Fig. 5.3. These points are shown in local sensor coordinates for a sphere of radius  $R = 10$  m. Fig. 5.3(a-b) illustrates

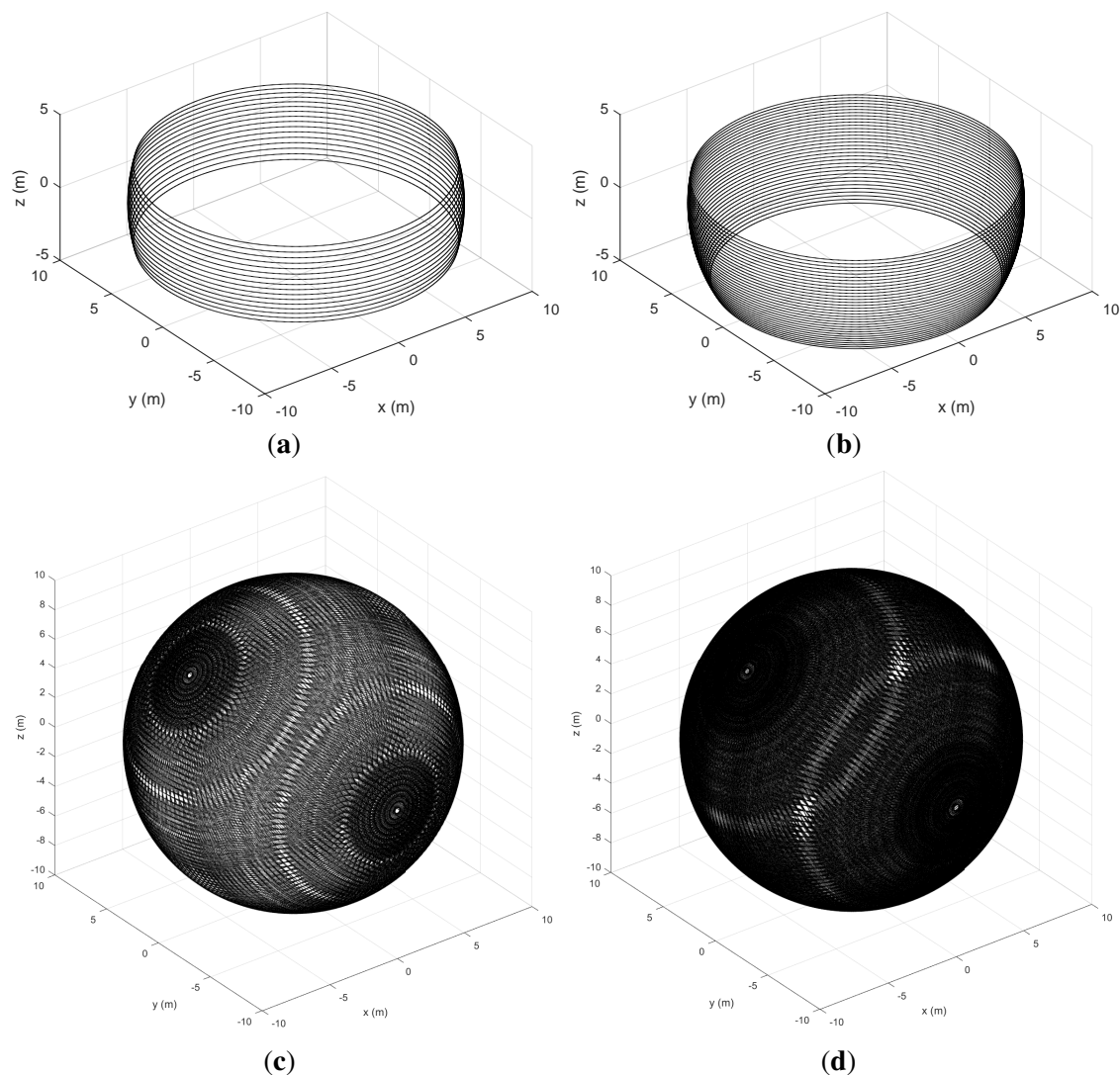


Fig. 5.3 Representation of lidar beams as sphere points: a) VLP-16, b) HDL-32, c) RMBL with tilting speed of  $120^\circ/\text{s}$  (29 frames), and d) RMBL with tilting speed of  $50^\circ/\text{s}$  (71 frames).

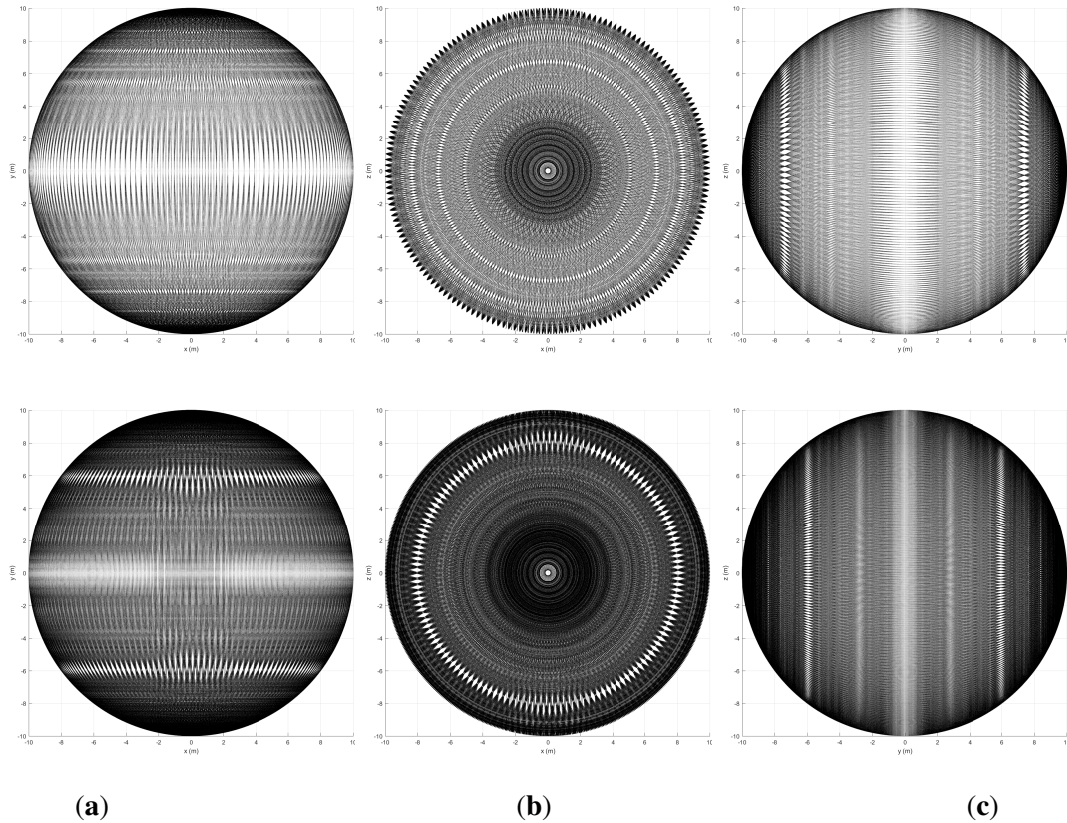


Fig. 5.4 Different views of the sphere points given by the RMBL with tilting speed of  $120^\circ/\text{s}$  (top) and  $50^\circ/\text{s}$  (bottom): XY plane (a), XZ plane (b), and YZ plane (c).

the inhomogeneous beam pattern of the Velodyne sensors as well as their differences in vertical resolution and FOV. Moreover, Fig. 5.3(c-d) shows points from RMBLs with different tilt speeds (i.e., different vertical resolutions). Points are distributed over the complete sphere but patterns due to the combination of the VLP-16 beams and the additional rotation are visible, especially in the lower resolution case. These patterns can also be appreciated on the lateral views of the spheres shown in Fig. 5.4. Furthermore, some pattern distortion is appreciable in the central vertical strip in Fig. 5.4(a-b), which correspond to a range of  $[-15^\circ, +15^\circ]$  around the extremes of the tilting motion. This can be explained because the eight VLP-16 transducers with positive  $\omega$  elevation values when  $\gamma = 0^\circ$  are overlapped with the eight transducers with negative  $\omega$  when  $\gamma = 180^\circ$ , and *vice versa*.

The translation of scan points onto planar surfaces in a synthetic environment is illustrated in Figure 5.5. The figure shows how the VLP-16 provides scarce information about the ground. This is improved in the HDL-32, but the maximum height of wall points is reduced. The RMBL offers denser data and a wider FOV for both ground and walls. Besides, it can be observed that RMBL point patterns on the target planes depend on the sensor orientation.

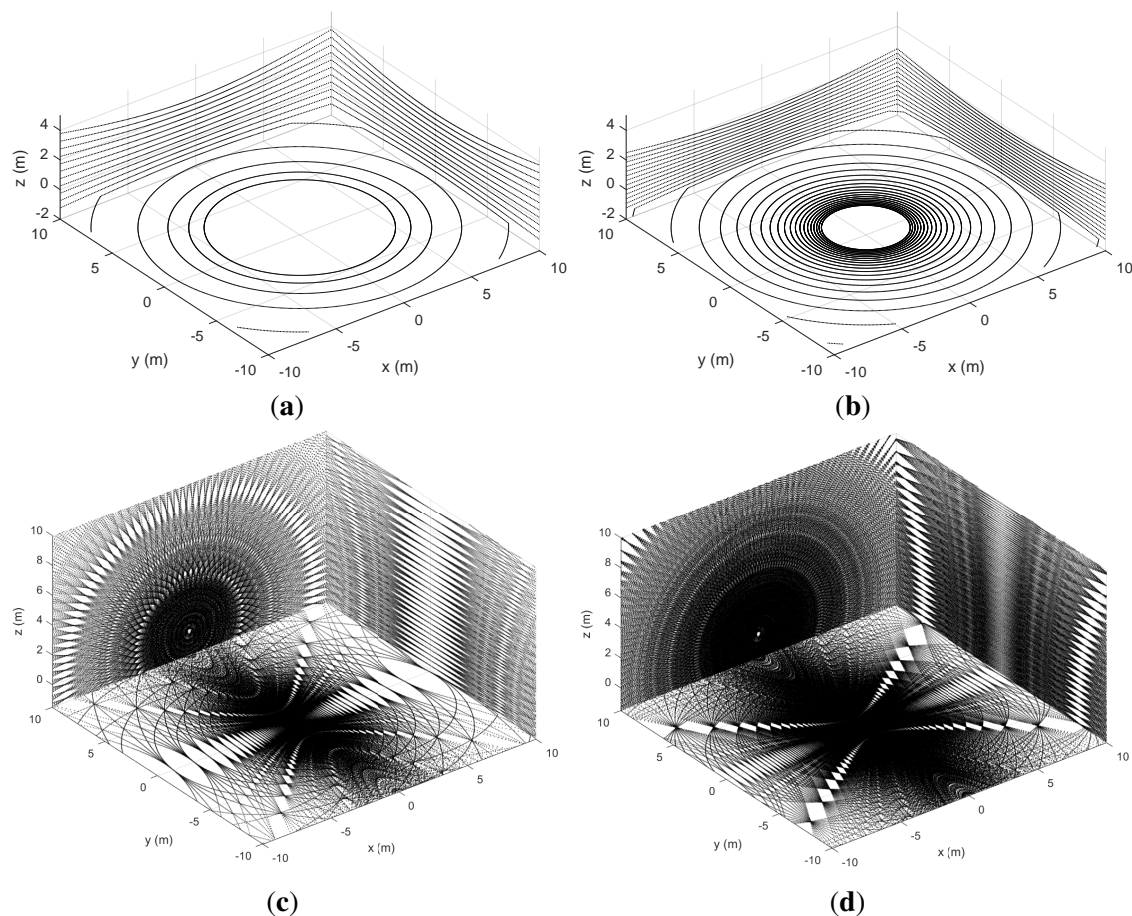


Fig. 5.5 Orthogonal plane points of lidar sensors: a) VLP-16, b) HDL-32, c) RMBL with tilting speed of  $120^\circ/\text{s}$  (29 frames), and d) RMBL with tilting speed of  $50^\circ/\text{s}$  (71 frames).

Thus, the wall perpendicular to the  $x$  axis is scanned with a higher density on the sides (i.e., similar to a good peripheral vision), whereas the wall perpendicular to the  $x$  axis is sampled with a higher density in the area that is close to the tilting axis. These differences should be considered when deciding on the sensor orientation with respect to the target surfaces in a particular application. Moreover, Figure 5.5(d) shows that a pattern of blind spots remains in spite of increasing the vertical scan resolution.

### 5.4.2 Sampling density

Beam density histograms for the RMBL with tilting speeds of  $120^\circ/\text{s}$  and  $50^\circ/\text{s}$  are presented in Fig. 5.6. For each case, the color scale has been normalized with respect to the maximum and minimum number of points per bin. The density increases notably in the polar regions (i.e., around the intersection with the rotation axis), which is a common trait with rotating 2D scanners [57]. The figure reflects that the slower scan offers more density, but also that the



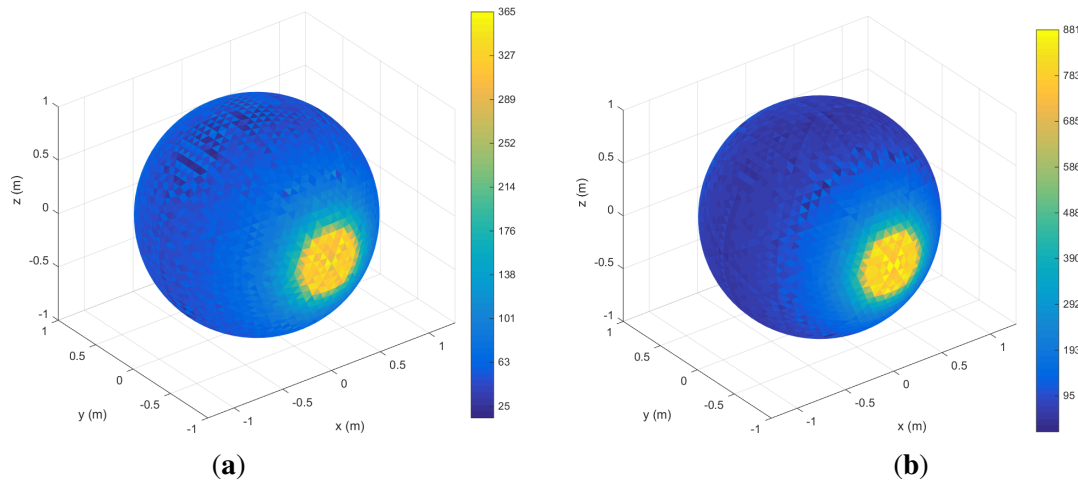


Fig. 5.6 2D histogram of beam density on the sphere: a) RMBL with tilting speed of  $120^\circ/\text{s}$  (29 frames), and b) RMBL with tilting speed of  $50^\circ/\text{s}$  (71 frames). Color bars indicate points/bin.

density distributions are similar regardless of tilting speed. However, a slightly darker and wider equator band in Fig. 5.6(b) would indicate a lower density in relation to the poles for the slower scan, which can be attributed to greater polar oversampling. The triangles with the maximum number of beams are in the polar regions, with 365 points for the  $120^\circ/\text{s}$  scan and 881 for  $50^\circ/\text{s}$ . In the lower resolution scan, the triangles with the minimum number of points (i.e., 25) lie around the equator band. When the tilt resolution is increased, the bins with less points (i.e., 95) are in two parallel bands closer to each polar region. These parallel dead zones are not completely eliminated by increasing tilt resolution, as can be seen in Fig. 5.4. The dead zones can be explained by the elevation gap between the VLP-16 transducers.

The same data is represented in Fig. 5.7 in a classical histogram graph. This figure indicates that the number of points is quite homogeneous with the exception of two groups of peaks that correspond to the polar regions. Again, the shapes of the histograms for both tilting speeds (i.e., elevation resolutions) are very similar. For the  $120^\circ/\text{s}$  speed, the mean is  $10.48 \text{ points}/\text{deg}^2$  ( $84.47 \text{ points}/\text{bin}$ ) with a standard deviation of  $6.98 \text{ points}/\text{deg}^2$  ( $56.24 \text{ points}/\text{bin}$ ); for  $50^\circ/\text{s}$ , the mean value is  $25.16 \text{ points}/\text{deg}^2$  ( $202.73 \text{ points}/\text{bin}$ ) with a standard deviation of  $16.62 \text{ points}/\text{deg}^2$  ( $133.92 \text{ points}/\text{bin}$ ).

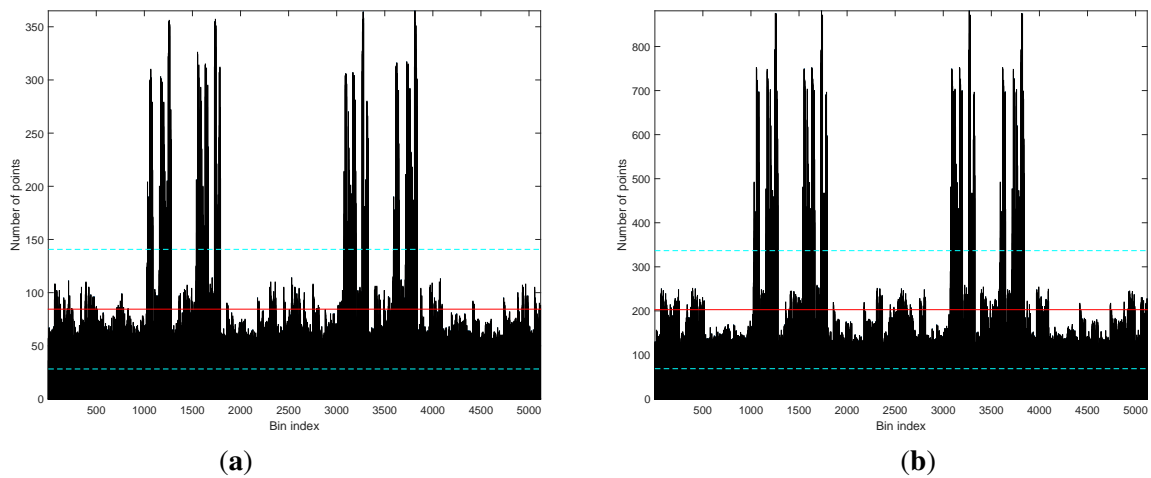


Fig. 5.7 Beam density histogram: a) RMBL with tilting speed of  $120^\circ/\text{s}$  (29 frames), and b) RMBL with tilting speed of  $50^\circ/\text{s}$  (71 frames). The red solid lines indicate the mean value and the blue dotted lines are the standard deviation.

## 5.5 Implementation of a portable tilting mechanism for a Velodyne VLP-16

The analysis methodology presented in Section 5.3 can be used to assess the scan measurement distribution of a real RMBL device and to establish a comparison with the ideal full-sphere values obtained in Section 5.4. An implementation of an RMBL consisting on a tilting multi-beam laser scanner has been developed in this work. This new device, named *Velomotion-16*, has been designed as a light portable platform based on the Velodyne VLP-16 scanner (see Figure 5.8).

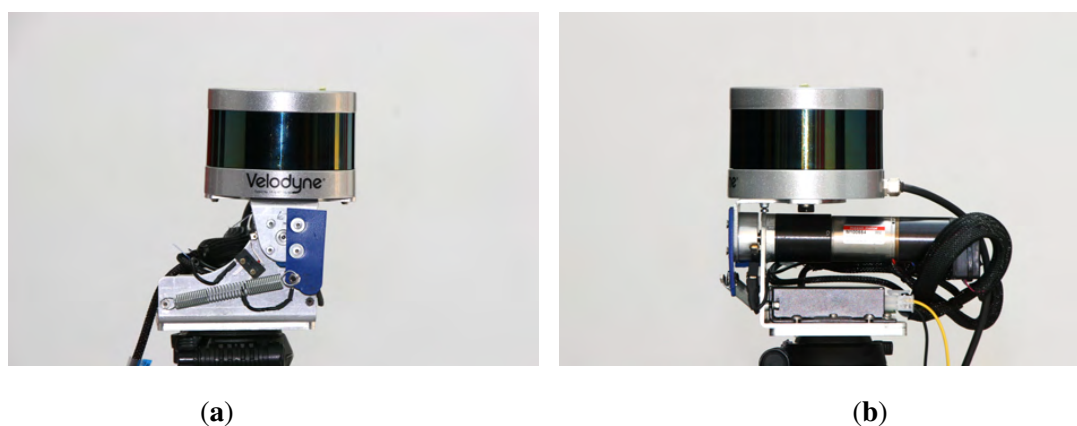


Fig. 5.8 Views of the Velomotion-16 RMBL sensor: (a) side; and (b) front.

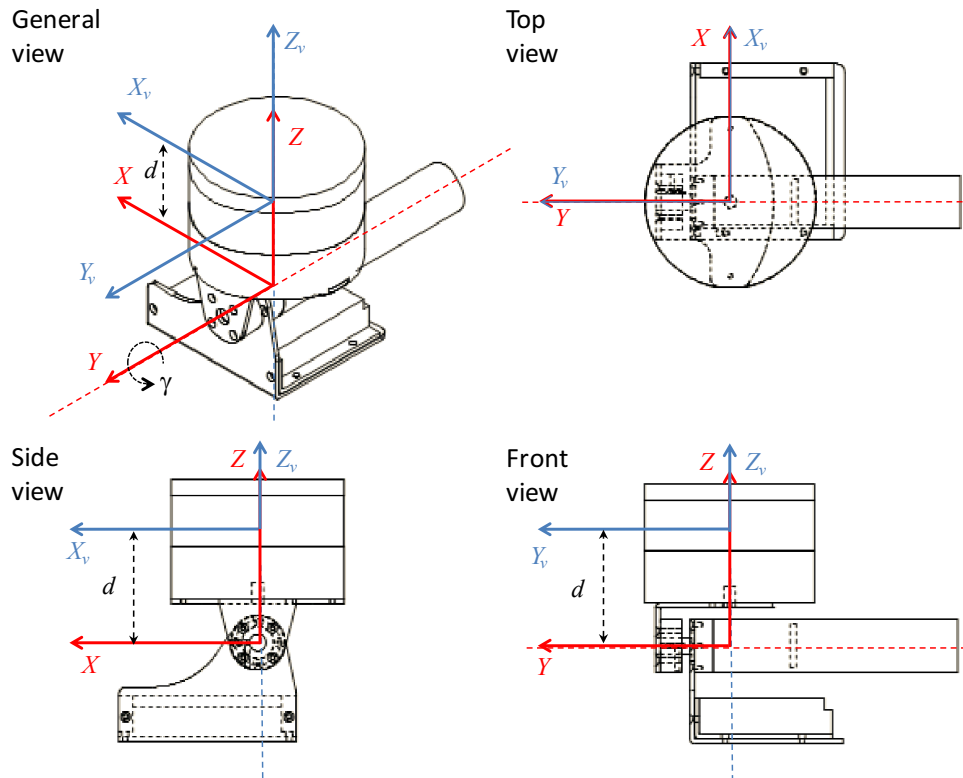


Fig. 5.9 *Velomotion-16* reference frames and tilting parameters. The VLP-16 local frame is represented in blue; the *Velomotion-16* local frame is represented in red.

### 5.5.1 Velomotion-16 system description

Several views of the *Velomotion-16* sensor design are presented in Figure 5.9. This figure shows the frames for VLP-16 frame and the *Velomotion-16* using the same notation as in Figure 5.2, including the tilting angle  $\gamma$  and the relative distance between the rotation mechanism and the optical center, which is  $d = 6$  cm.

The main specifications of the tilting platform are presented in Table 5.2, where some parameters are inherited from the constituent VLP-16 sensor. In this device, there is a mechanical limitation regarding the additional DOF, which is in the range  $[-45^\circ, 0^\circ]$ . This means that the vertical FOV is asymmetrical with respect to the horizontal plane, as it is  $[-60^\circ, 15^\circ]$  in the forward direction and  $[-15^\circ, 60^\circ]$  backwards.

Table 5.2 Specifications of Velomotion-16, as used in the case study (VLP-16 device included). When values are inherited from the VLP-16, this is indicated.

Range	1 m to 100 m (VLP-16) + offset $\leq d$
Accuracy	$\pm 3$ cm (VLP-16)
Data Rate	300,000 points/s (VLP-16)
$d$	6 cm
Tilting range	$[-45^\circ, 0^\circ]$
Tilting speed	$0.05^\circ/s$ to $56.25^\circ/s$ (programmable)
Vertical FOV	$75^\circ$ : $[-60^\circ, +15^\circ]$ (forwards), $[-15^\circ, 60^\circ]$ (backwards)
Vertical Resolution	uneven
Mean vertical resolution	$5.2^\circ \cdot 10^{-3}$ to $0.59^\circ$ (programmable)
Horizontal FOV	$360^\circ$ (VLP-16)
Horizontal Resolution	$0.1^\circ$ to $0.4^\circ$ (programmable) (VLP-16)
Size	105 mm width $\times$ 95 mm height $\times$ 165 mm depth
Weight	1.9 kg (+0.7 kg wires)

The range of Velomotion-16 is inherited from that of the VLP-16, but it is affected by a positive offset not greater than  $d$ . The actual limits of the sensing range depend on  $(\alpha, \omega, \gamma)$ . Given that a range measurement  $P$  (see Figure 5.2) is:

$$P = \sqrt{x^2 + y^2 + z^2}, \quad (5.7)$$

then, the actual minimum and maximum range values for  $P$  can be computed using Equations (5.4) with  $R = 1$  m and  $R = 100$  m, respectively.

From a mechanical standpoint, the portable tilting platform consists of two L-shaped links with a rotational joint. The base link has been designed to accommodate the controller card and the motor-gear-brake set and includes two switches to restrict the displacement. The VLP-16 support link has been designed to be lightweight, to achieve a small  $d$  value, and to make the  $Y$  and the  $Y_v$  axes parallel, as in Figure 5.2. A cylindrical coupling piece joins the output axis from the reduction gear with the VLP-16 support. Furthermore, a spring avoids the clearance between the base link and the VLP-16 support.

A general overview of the system architecture is shown in Figure 5.10. The tilting motion is achieved by an electronically communicated (EC) brushless motor with encoder and brake and an EPOS2 controller, both by Maxon (Sachseln, Switzerland). Two 12 V batteries are used to provide 12 V power to the system (including the VLP-16) and 24 V to the brake. Motion control is performed through a trapezoidal profile in which speed, acceleration and



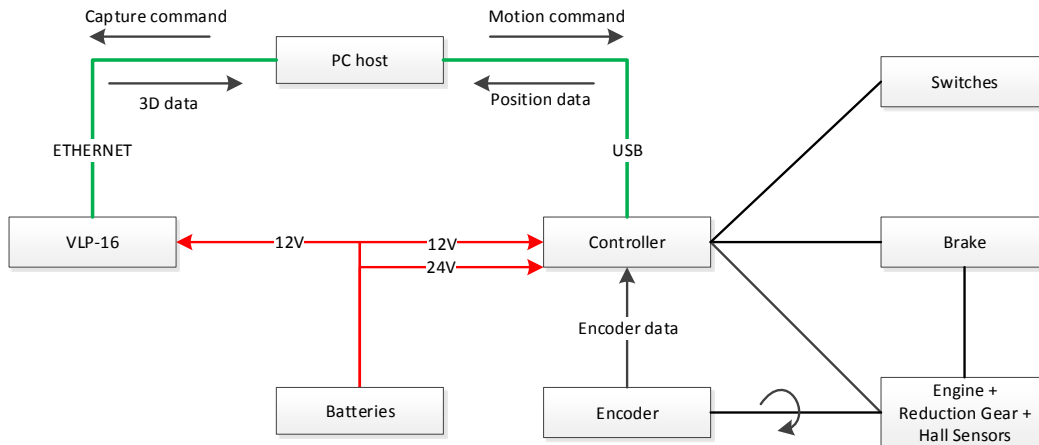


Fig. 5.10 Velomotion-16 system architecture.

deceleration can be specified by the user to produce a particular scanning resolution. The fastest scan is achieved by setting the tilting speed to  $56.25^\circ/\text{s}$ , which corresponds to eight VLP-16 scans. Conversely, high density scans can be obtained by programming slower tilt speeds, which can be as low as  $0.05^\circ/\text{s}$ .

As illustrated in Figure 5.10, the PC host sends capture commands and receives 3D data as ROS messages from the VLP-16 via Ethernet. Moreover, the PC sends the goal position of the motion profile and receives the current angle from the tilting platform through a USB connection. A ROS driver has been developed to synchronize consecutive VLP-16 scans with the corresponding tilt angles in order to generate a dense point cloud.

### 5.5.2 Analysis of the scan measurement distribution for Velomotion-16

The effects on the scan measurement distribution produced by a particular sensor construction can be identified by applying the proposed analysis methodology. This is illustrated in Figure 5.11 for Velomotion-16, which has a limited tilting range. The results in the figure correspond to a tilting speed of  $50^\circ/\text{s}$ .

The complete FOV originated by the tilting range, which can be clearly appreciated in Figure 5.11a, has a downwards orientation in the negative direction of the  $X$  axis (i.e., backwards). The hollow sphere patterns are similar to those of the ideal sensor with the exception of the scan lines in the extremes of the tilt range, which are sparser because they are not overlapped. This sparseness can be appreciated in the forward floor plane points of Figure 5.11b. Orthogonal plane points also reveal the efficacy of Velomotion-16 to scan ground points in the backward direction, whereas the forward direction is more appropriate to capture higher vertical structures.

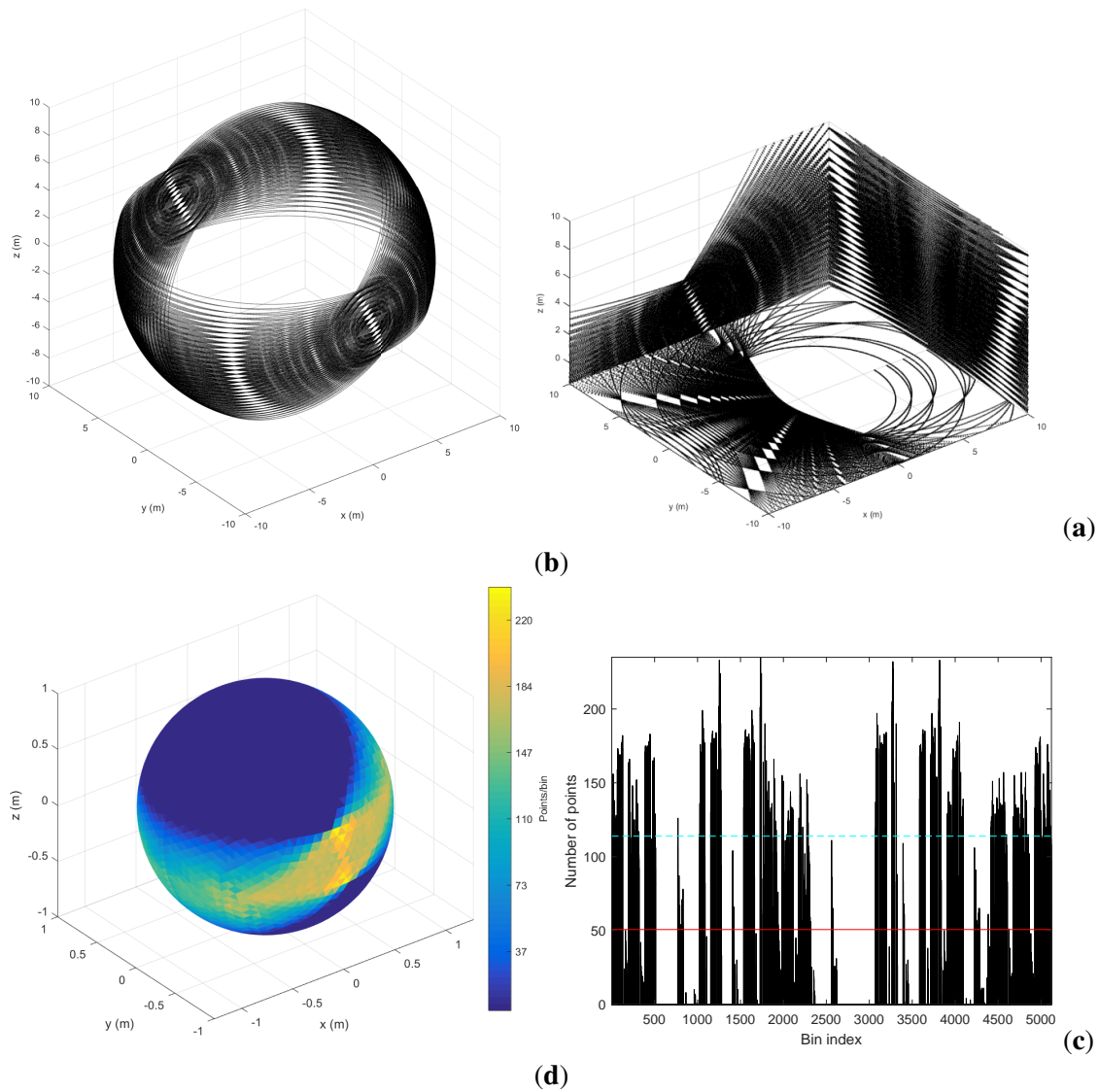


Fig. 5.11 Analysis of 3D scan measurement distribution for *Velomotion-16* with tilting speed of  $50^\circ/\text{s}$  (17 frames): (a) hollow sphere points; (b) orthogonal plane points; (c) point density on the sphere (dark blue means no measurements); and (d) point density as a histogram.

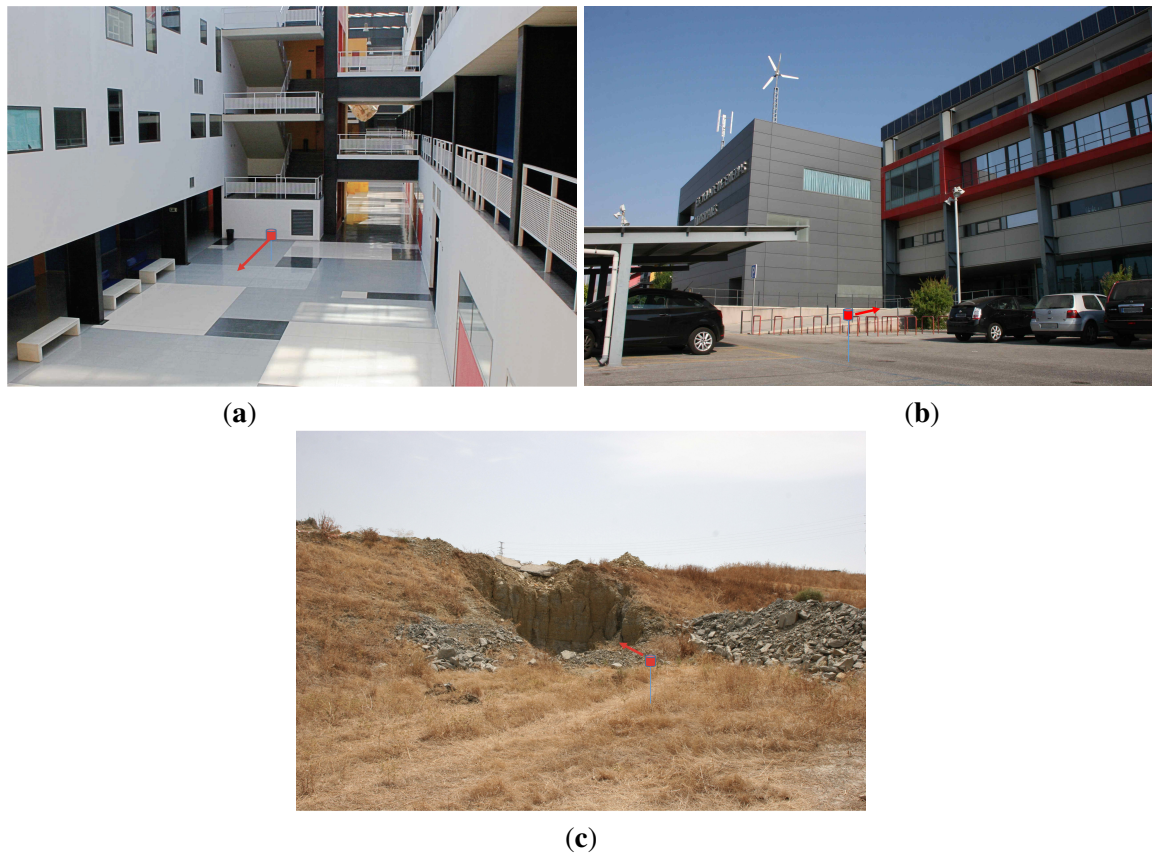


Fig. 5.12 Photos of experimental scenes: a) Building hall (indoor), b) Parking area (urban), and c) Quarry (terrain). The approximate position and forward scan direction of the Velomotion-16 sensor is indicated by a red arrow.

The point density histograms in Figure 5.11c,d also indicate unscanned regions as well as a slighter scan density in the extremes of the tilt range. The mean point density has been reduced from 25.16 points/deg<sup>2</sup> (202.73 points/bin) in the ideal full-sphere to 6.29 points/deg<sup>2</sup> (50.68 points/bin) in the Velomotion-16 sensor. Similarly, the standard deviation has changed from 16.62 points/deg<sup>2</sup> (133.92 points/bin) to 7.85 points/deg<sup>2</sup> (63.22 points/bin).

## 5.6 Discussion of example scans

The purpose of this section is to illustrate 3D lidar data from actual scans obtained in three representative scenes of indoor and outdoor environments: a building hall, an outdoor parking area in a urban setting, and a quarry with irregular terrain, respectively (see Figure 5.12). Actual scans have been obtained by Velodyne's VLP-16 and HDL-32 MBLs as well as for the *Velomotion-16* RMBL with two different tilting speeds (i.e., vertical resolutions): the

Table 5.3 Sensor performance in example scans.

	scan time (s)	$\gamma$ speed ( $^{\circ}/s$ )	Indoor: points	Urban: points	Terrain: points
Velomotion-16 (slow)	42.05	1.07	11,348,103	6,833,873	6,269,304
Velomotion-16 (fast)	0.80	56.25	201,851	158,351	125,237
VLP-16	0.10	-	27,998	20,141	16,954
HDL-32	0.10	-	68,080	58,875	51,362

fastest scan speed given by the sensor (i.e.,  $56.25^{\circ}/s$ ) and a slow high resolution speed of  $1.07^{\circ}/s$ . The sensors have been placed on a tripod at a height of about 1.2 m.

Table 5.3 summarizes sensor performance by presenting the scan time for each case, as well as the resulting number of points for each scenes. The total number of points given by the Velomotion-16 is substantially larger in the indoor environment due to out-of-range measurements in the urban environment. This difference is not so important in the Velodyne sensors, as their FOV is very limited in the upwards direction. The fast Velomotion-16 requires 0.8 seconds (i.e., eight times the Velodyne scan time) to capture eight VLP-16 scans, which produces a greater number of points (i.e., 7.21 times and 7.86 times, for the indoor and urban scenes, respectively) than a single VLP-16 scan as well as a wider FOV. Furthermore, the fast Velomotion-16 also improves the number of points and the FOV of the HDL-32, which is a considerably more expensive sensor. By adjusting the Velomotion-16  $\gamma$  speed to a slower value, the resulting number of points, and the subsequent data density, can be greatly improved, as indicated by the numbers given by the slow case in the Table.

The results for these experiments are presented in Figures 5.13-5.15, where the color grading represents elevation. These illustrations confirm the improvement in data density and FOV provided by the Velomotion-16 with respect to both Velodyne lidars. The particular mechanical rotation limits of the Velomotion-16 favor a denser resolution of the floor in the forward direction and a better measurement of higher objects in the backwards direction. Interestingly, even the fast Velomotion-16 offers a high point density in the immediate floor area. In the quarry scene, the forward scan direction provides a detailed scan of the ground terrain and the excavated wall, especially in the slow scan.

As for the particular scan patterns presented in Figure 5.5, these can also be appreciated in the floor and vertical walls of these actual scans. In general, these patterns become difficult to identify when the resolution is increased in the fast Velomotion-16, but the pattern of small blind spots can be seen on the wall at the center-left part of Figure 5.13(d), the floor at the bottom-right side of Figure 5.14(d), and the bottom-left diagonal and the top-right area of Figure 5.15(d).



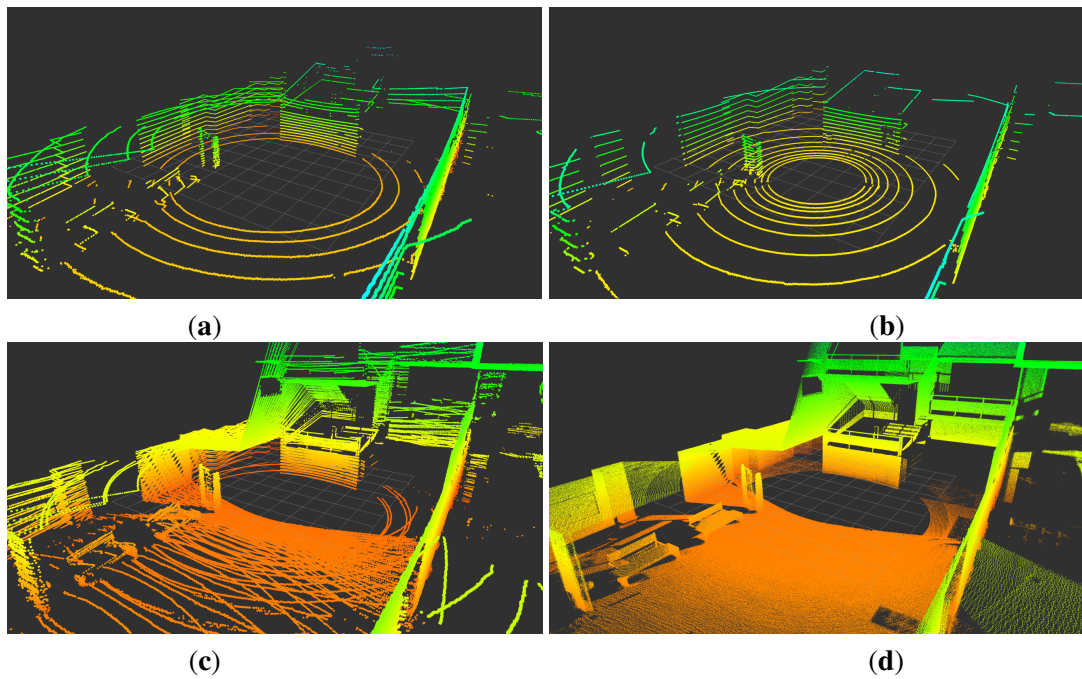


Fig. 5.13 Scans from an indoor scene: a) VLP-16, b) HDL-32, c) Fast Velomotion-16, and d) Slow Velomotion-16.

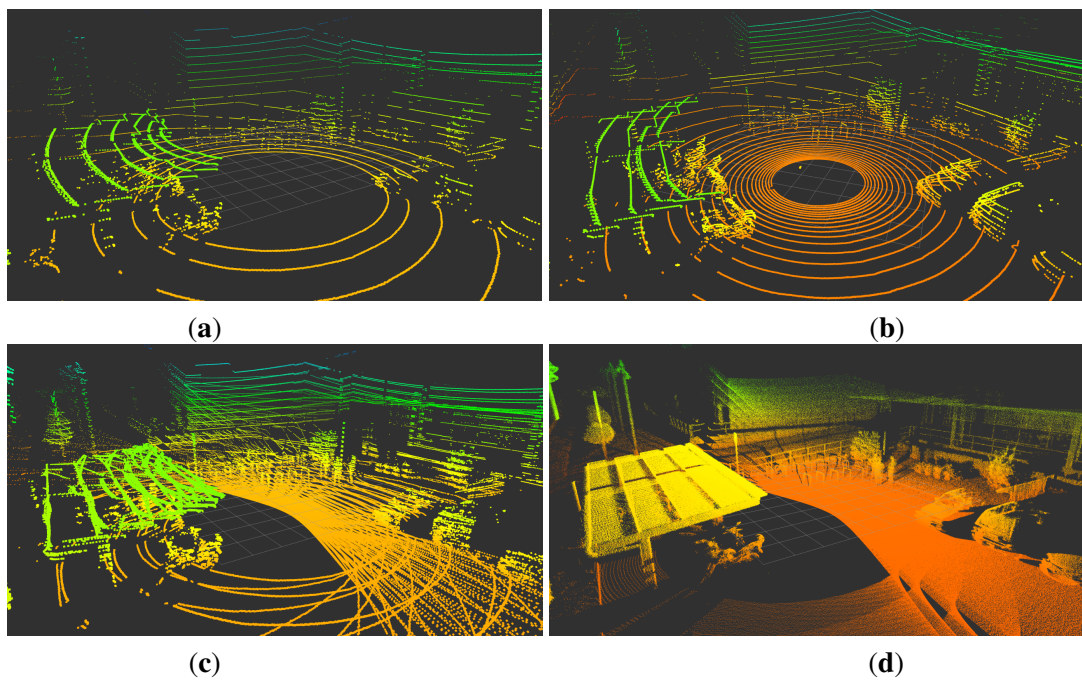


Fig. 5.14 Scans from the urban scene: a) VLP-16, b) HDL-32, c) Fast Velomotion-16, and d) Slow Velomotion-16.

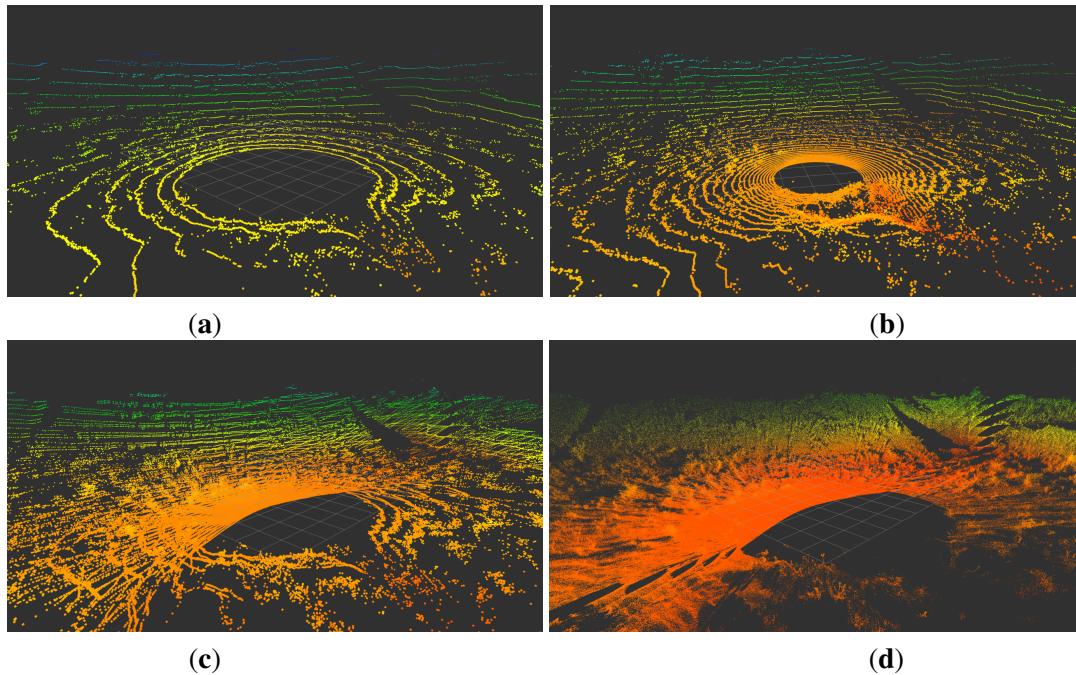


Fig. 5.15 Scans from an outdoor terrain scene: a) VLP-16, b) HDL-32, c) Fast Velomotion-16, and d) Slow Velomotion-16.

## 5.7 Summary

This chapter has addressed the analysis and development of a RMBL sensor, a new type of customized 3D rangefinders built by adding a rotation mechanism to a commercial MBL. Other recently published examples using VLP-16, the most affordable and lightest MBL by Velodyne, indicate that the RMBL configuration has the potential to become a common solution to get low-cost, rapid, and full-3D high resolution scans, as has happened with customized RSML during the last fifteen years. However, contrary to RSBLs, the additional DOF in RMBLs causes multiple scan beams to overlap creating complex scanning patterns.

Particularly, this chapter has proposed a simulation-based analysis of 3D scanning patterns, which has been applied to investigate the complex scan measurement distribution produced by the RMBL configuration. With this major purpose, novel contributions offered in the chapter include the following: a comparison, both qualitative and quantitative, of general scan patterns produced by an ideal VLP-16-based RMBL with other 3D scan alternatives (i.e., SMBL and MBL), the design and implementation of a new RMBL consisting of a portable tilting platform for Velodyne VLP-16 scanners, and a discussion of actual example scans obtained with the new device.

Qualitative analysis evidences particular sampling patterns provoked by the RMBL configuration. Most of these patterns are difficult to appreciate when the resolution of the

additional rotation is increased with slower scans. Nevertheless, rows of characteristic small blind spots remain visible even with high resolution rotation. Apart from that, similarly to RSBLs, the measurement density has focal points in the rotation axis, which has to be considered when placing the sensor for a particular application.

The chapter has also proposed a new tilting RMBL sensor based on the VLP-16. This is a compact and portable device that weights a total of 2.6Kg. Experimental example scans obtained by this device have illustrated a practical implementation of the RMBL configuration. These scans have shown that even a fast tilt (in less than one second) of the VLP-16 provides an environment description that can be considerably richer (both in FOV and number of points) than that of the HDL-32. A much higher level of detail can be appreciated in scans taken in less than a one-minute span. Given the significant difference in price between the VLP-16 and the HDL-32 (not to mention other high-end 3D lidars), these results support the feasibility of customized RMBLs in stop-and-go applications demanding affordable and compact high-resolution point clouds.

In the future, the proposed simulation methodology can be applied to assess the effect of particular rotating mechanisms with respect to these reference measure distributions. Furthermore, it will also be interesting to analyze the effect of rotating other MBLs, since there is an increasingly active MBL market for new compact and cost-effective devices fostered by automotive applications (e.g., the 32-beam Ultra Puck VLP-32C by Velodyne [109]).

The scientific production of this chapter is listed below:

- [66] Sensors. Analysis of 3D Scan Measurement Distribution with Application to a Multi-Beam Lidar on a Rotating Platform. 2018. (Journal Q1).



UNIVERSIDAD  
DE MÁLAGA



# Chapter 6

## Conclusion and future work

### 6.1 Conclusion

Intelligent scene understanding is a complex process which is used in many applications like autonomous navigation. Accurate and high resolution 3D information about the scene can be obtained as a 3D point cloud by sensors such as cameras or 3D scanners. This large amount of raw data, must be transformed into an effective representation in order to help posterior processing such as object detection or navigation control. This thesis work has offered contributions to intelligent scene understanding of unstructured environments from 3D point clouds, in particular new algorithms for point cloud classification, object segmentation, navigability assessment, and the development of a tilting MBL.

Classification of spatial shape features based on PCA is a widely used constituent process in recent scene processing methods [49][55]. In this sense, geometric features can describe the shape of regions and objects for later contextual classification [127]. Therefore, this thesis has been focused on improving the effectiveness, both in computational load and accuracy, of supervised learning classification of spatial shape features (i.e., tubular, planar or scatter shapes) obtained from covariance analysis [47]. While many authors define categories such as vertical or horizontal orientation [32], in this work geometric features have been identified regardless of their orientation, which extends applicability to unstructured environments.

Since computing neighborhood for 3D points is a costly process for classification, this thesis has also proposed a novel concept of a general framework for implementing and comparing different supervised learning classifiers with a voxel-based neighborhood computation. This way, based on a regular 3D grid, features are computed for each non-overlapping voxel, and all points in the same voxel are assigned to the same class, resulting in a GFVM. The feasibility of this approach has been evaluated by implementing a NN method, in particular a multi-layer perceptron-based model, and three other supervised learning classifiers found in

scene processing methods (SVM, GP, and GMM) as well as five alternative feature vector definitions based on principal component analysis for scatter, tubular and planar shapes. Classification performance metrics and processing time measurements have confirmed the benefits of the NN classifier and the feasibility of the voxel-based neighborhood approach for terrestrial lidar scenes, also with respect to point-wise neighborhood. Results have also shown that the choice of suitable features can have a dramatic effect on the performance of classification approaches. The method has been successfully applied to real scans from both, natural and urban environments from two different 3D rangefinders.

This work has also proposed applications based on GFVM towards the characterization of a 3D scene. In particular a segmentation method and a generation of an occupancy grid specially designed for natural environments. Both algorithms are based on a ground segmentation method which also extends the GFVM representation by adding a new attribute to the voxels, apart from their geometric class. The proposed ground segmentation algorithm considers slopes, non-flat areas, and isolated areas, which are commonly presented in unstructured scenes, as flat ground assumptions in methods for urban scenes, like plane fitting [125] or height thresholds [53], do not apply. To identify ground voxels, the algorithm consists of a set of refining rules whose optimal sequence has been determined from experimental analysis. Once the ground is segmented, a double-segmentation method is performed, first by isolating the objects, and then splitting them into regions with same geometric class. The proposed algorithm has been tested in urban and natural environments presenting better results in comparison with other methods from the literature in natural environments. In addition, an experiment has been executed to conclude which size of voxel provides the best result. Besides, a novel metric has been introduced to quantify the accuracy in the segmentation through a comparison between each segmented point cloud and the same one, manually labeled. Furthermore, the author has proposed a method for 2D occupancy grid generation which is efficient for local navigation in natural environments. The proposed 2D occupancy grid represents accessible areas by considering the attributes and the height of each voxel over the ground level in relation to vehicle dimensions. As a result, an effective map is obtained where the objects in the scene are classified in geometric classes and finally labelled as traversable or non-traversable area.

The methodologies proposed in this work may be fully integrated in an mobile robot with an embedded terrestrial lidar scanning (TLS) or any other technology to capture dense 3D point clouds (see Section 2.1). For semi-autonomous systems, the geometric and semantic information can help the teleoperator to plan the route, such as in SAR missions where the operator can not access to the search area. GFVM may contribute in planetary exploration where the robot aims to capture and map a big amount of information into an optimal data

structure to save memory and bandwidth. In addition, for fully autonomous vehicles, the occupancy grid becomes an appropriate input for the path planning algorithm. Moreover, all the descriptors deducted from the 3D scene can be processed by a recognition algorithm to identify the objects, for instance, in a guiding system for blind people. Registration of consecutive frames and mapping problems are widely used, also in urban scenes, with the ICP algorithm which is point-based and hence a costly process, so the GFVM can contribute to reduce the computational cost. Consequently, an optimized SLAM algorithm may be produced based on GFVM. Furthermore, these methods may be partially integrated in aerial vehicles, such as drones, with embedded airborne lidar scanning, to describe forest canopy or detect fraudulent constructions. Hence, although the initial purpose of this thesis was to contribute to long range exploration in natural environments for ground mobile robots, the work is suitable for a wide variety of applications, in fact, the work also fits to urban, indoor, or aerial scenes.

This thesis has also addressed the design, development and analysis of novel 3D laser rangefinders based on commercial multi-beam sensors. In particular, the thesis has proposed a tilting rotation to the Velodyne Puck with the purpose of increasing density, which is made more homogeneous, as well as the vertical field of view. The point clouds generated by multi-beam scanners have a sparse and inhomogeneous density which affects the results of typical processing techniques [111]. This work has presented a compact, affordable and light design which can be used as a portable sensor. This design has been implemented as the new *Velomotion-16* 3D sensor which improves the limitations from multi-beam scanners. In addition, this work has proposed a methodology to evaluate 3D scanning patterns, which has been applied to investigate the scan distribution.

Summarizing, this thesis has presented contributions towards intelligent scene understanding of unstructured environments from 3D point clouds. The major contributions include new methodologies for point cloud geometric classification, object segmentation, navigability assessment and the development and qualitative analysis of a tilting MBL. Since the 3D data imply large data structures and hence a high computational cost, the proposed solution considerably reduces the computational cost with respect to other 3D mapping methods by reducing the amount of data to process. This thesis has also presented a low cost solution, based on a new developed affordable 3D scanner that improves the density and homogeneity of the resulting point cloud. All the proposed solutions has been successfully evaluated with real scans of urban and natural scenes provided by two different models of laser rangefinder. These methods can be applied to semi-autonomous robots in natural environments or in a SAR scene.

## 6.2 Future work

The work presented in this thesis could be expanded with the following contributions:

- **Integration of the whole method in an autonomous vehicle**

Integration of all the proposed methods, starting from the capture of 3D point cloud with the new laser range device, through the voxelization, geometric classification, segmentation and occupancy grid generation in order to a step forward toward the path planning and autonomous navigation. In ROS there are already implemented path planning algorithms to test with the generated occupancy grids from the environment. This work is planned to be integrated into the Rambler robot (developed in the same project which is this thesis associated to).

- **Parallel version of the proposed classification method with multi-core computers or GPU**

One additional advantage of processing each non-overlapping cell by using points from only that same cell is that this favors parallelization [94]. Developing a parallel version of the proposed method to improve online classification time with multi-core computers or GPU.

- **Incremental update of classification with consecutive scans**

Apart from the incremental mapping with consecutive scans, it will be also interesting to adapt the method for incremental update of classification results instead of processing again repeated voxels. The update may include the voxelization of new points, the geometric features extraction, and the classification of those new voxels.

- **Object classification in the scene**

Since the segmentation step already splits the scene into objects and geometric regions inside each object, several geometric and statistical features can be extracted from both objects and regions in order to classify objects automatically. Preliminary work have been already made obtaining promising results identifying trees, cars, lamp posts, buildings, pedestrian and bushes.

- **Improve the Velomotion-16 from prototype to finished product**

The new laser-rangefinder is a prototype which lacks some improvements that could be implemented producing a finished product. In particular, an user interface for the configuration, an interface for the ROS environment, mechanical optimization and the integration of efficient and compact batteries

# Resumen de la tesis doctoral

## Introducción

El reconocimiento automático del entorno es un proceso complejo que se usa en robots con tareas autónomas tales como la navegación o manipulación. En vehículos autónomos, la identificación precisa de terreno navegable y obstáculos es crucial para la navegación segura. Además, nuevos retos de aplicaciones para vehículos autónomos, como la búsqueda y rescate o la exploración planetaria, tienen lugar en entornos naturales o desestructurados. Los sensores exteroceptivos que llevan incorporados, como cámaras o escáneres 3D, pueden producir información precisa y de alta resolución sobre el entorno inmediato, pero esta gran cantidad de datos en bruto puede ser difícil de interpretar para un proceso de toma de decisiones. Por este motivo, la información compleja de estos sensores debe ser procesada de tal modo que pueda servir al control de navegación. El principal propósito de esta tesis es ofrecer contribuciones al desafiante problema de entendimiento de escenas en entornos desestructurados basadas en nubes de puntos 3D. La mayor contribución incluye nuevas soluciones para la clasificación de nubes de puntos, segmentación de objetos, estimación de navegabilidad, y el desarrollo y análisis cualitativo de un dispositivo lidar multi-haz giratorio.

Debido a la gran cantidad de datos capturada de la escena 3D, la enorme nube de puntos puede ser transformada en una representación más eficiente para reducir el tiempo de ejecución en el procesamiento de extracción de características o posteriores aplicaciones como la clasificación de objetos o la planificación de trayectorias. Hay tres grupos principales de técnicas: *i*) reducción de datos en la nube de puntos, por ejemplo, mantener sólo los del área local en el frente [86] o descartar información redundante [57], *ii*) transformación de la información en un modelo 3D como mapa de elevación [92][56] que reconstruye el terreno en dos dimensiones y media perdiendo información geométrica relevante de la escena, o mapas de vóxeles [17] que evitan este problema, o *iii*) extracción de un modelo bi-dimensional (2D) tal como un mapa de alturas [29] o un mapa de ocupación [103] [23]. Esta tesis propone un procesamiento de vecindario basado en vóxel donde los puntos en cada vóxel no solapado en una cuadrícula uniforme son procesados a la vez para extraer

características 3D relevantes. Esta solución basada en vóxel reduce significativamente el procesamiento de datos en comparación con las técnicas basadas en puntos.

La clasificación de características geométricas primitivas es extensamente usada como paso inicial en problemas de interpretación de alto nivel de escenarios [116]. Por ejemplo, la clasificación de puntos en categorías geométricas gruesas como vertical y horizontal, ha sido propuesta como la primera capa de una metodología jerárquica en el proceso de escenas urbanas complejas [32]. Además, la clasificación de puntos escaneados previa a la segmentación es útil para procesar objetos con bordes confusos, como el suelo, vegetación y copas de árboles [8]. En este sentido, las características de forma espacial pueden describir la forma de objetos para la posterior clasificación contextual [127]. Por tanto, la clasificación de características de forma espacial basada en análisis de componentes principales es un proceso elemental en métodos recientes de procesamiento de escenarios [125][18][38][49][55]. Por todo ello, esta tesis se centra en mejorar la eficiencia, tanto de carga computacional como de precisión, de la clasificación de aprendizaje supervisado de características de forma espacial (i.e., forma tubular, plana, o difusa) obtenidas a partir de análisis de covarianza [47].

Tras el procesamiento de datos de la escena en bruto, como la voxelización y la clasificación geométrica, se pueden aplicar segmentación y análisis de navegabilidad. Un método de segmentación se usa para aislar objetos individualmente de la escena como un paso previo a la clasificación o reconocimiento semántico de los mismos. Existen muchos trabajos dedicados a resolver el problema de la segmentación de imágenes 2D, y más recientemente el desafío también se ha aplicado a las nubes de puntos 3D, ya que los sensores 3D están siendo cada vez más usados. En [26], algunos algoritmos de segmentación se han desarrollado para nubes de puntos densas y dispersas, lo cual depende del dispositivo de escaneo, concluyendo que los algoritmos que primero extraen el suelo ofrecen resultados más precisos en el siguiente proceso de segmentación. Algunos trabajos empiezan el proceso extrayendo el suelo ajustándolo a un plano [126] [125] o usando un umbral de altura [122]. Después, los puntos que no son suelo se agrupan en objetos o segmentos dependiendo de la distancia entre puntos usando un algoritmo basado en grafo como el *Normalized Cut* o *Ncut* [101] [122] y el algoritmo de Felzenszwalb and Huttenlocher (FH) [31] sobre la imagen de profundidad [126] o sobre la nube de puntos 3D [102]. Otros métodos realizan la segmentación basada en la similaridad del vector normal calculado en cada punto teniendo en cuenta los puntos vecinos. En [89], la normal de cada punto se calcula con la información de los  $k$ -Vecinos más cercanos. Para reducir la estimación de la normal en cada punto, en [125], los puntos son primero agrupados si están más cerca que un umbral de distancia, esto se llama técnica de región creciente, y después esos clústers se agrupan en base a la similaridad de la normal. El mayor inconveniente de estos enfoques de segmentación basados en punto es la gran cantidad

de memoria y tiempo de ejecución que consumen ya que tienen que procesar cada punto de la nube, y por tanto puede ser incompatible con aplicaciones de tiempo real. Otro conjunto de técnicas para segmentación 3D son las técnicas basadas en grupo, las cuales intentan resolver el problema de consumo de tiempo de las técnicas basadas en punto reduciendo la escena a un modelo de vóxel. Normalmente la escena se discretiza en cubos 3D con tamaño constante como una cuadrícula 3D, pero estos vóxeles también pueden posicionarse dinámicamente alrededor de grupos de puntos [37]. Una vez que el suelo se ha extraído, los objetos se aíslan mediante la fusión de vóxeles adyacentes [25]. Otros autores no extraen el suelo como paso previo, sino que implementan la segmentación basada en información geométrica como las características geométricas locales (autovalores de la matriz de covarianza, o PCA) [97] o además de esto, también basada en la altura y vector normal [38] sobre el vecindario a nivel de punto o de vóxel. El mayor problema de estas técnicas es que asumen que el suelo es plano y continuo, como en entornos urbanos. Sin embargo, la detección de suelo es compleja para entornos desestructurados donde el suelo contiene grandes inclinaciones y áreas no planas.

Los dispositivos comerciales para obtener datos 3D densos y precisos son caros, pesados y/o de gran consumo energético. La mayoría de soluciones para minimizar el coste de un dispositivo escáner 3D consiste en rotar un escáner láser 2D en cualquier eje [59] [63]. Sólo algunos autores, [50] y [72], añaden una rotación extra a un dispositivo 3D en sí mismo (Velodyne) para aplicaciones de mapeo. Los sensores lidar Velodyne están siendo cada vez más usados en muchas aplicaciones de escaneo 3D. Sin embargo, las nubes de puntos generadas por escáneres multi-haz, como Velodyne, tienen densidad dispersa y heterogénea. Esta mencionada heterogeneidad en las nubes de puntos afecta sustancialmente al resultado de las típicas técnicas de procesado, como la búsqueda del mejor punto de correspondencia para emparejamiento de escáneres [111]. Ya que la densidad aumenta alrededor del nuevo eje de rotación, [121] implementa no sólo una como normalmente, sino dos rotaciones adicionales, de cabeceo y balanceo, para hacer la densidad uniforme. Esta tesis propone añadir una rotación de cabeceo a un Velodyne Puck con el propósito de incrementar la densidad y la homogeneidad, y también el campo de visión vertical. En particular, se presenta un diseño ligero y compacto que puede ser usado como sensor portátil. El diseño ha sido implementado como el nuevo sensor 3D *Velomotion-16*.

Esta tesis aborda el desafío de robótica del entendimiento de escenario considerando la clasificación de nube de puntos, segmentación en terreno natural y otros objetos de la escena, y el análisis de navegabilidad. Además, esta tesis explora las posibilidades ofrecidas por la agregación de un grado de libertad a un sensor lidar multi-haz para obtener de manera asequible y rápida escaneos 3D de alta resolución.



## Contribuciones

El objetivo de esta tesis es contribuir con nuevas metodologías y desarrollos al entendimiento inteligente de escenarios en entornos desestructurados basados en información de sensor lidar 3D. La tecnología de este sensor, que produce nubes de puntos densas, puede ser especialmente apropiada para aplicaciones nuevas en entornos naturales o desestructurados, tales como la búsqueda y rescate y la exploración fuera de carretera. Esta área de investigación es un reto que engloba disciplinas desde el diseño de sensor a la inteligencia artificial y *machine learning*. Las contribuciones incluyen nuevas metodologías para la clasificación de características espaciales, segmentación de objetos y cálculo de navegabilidad en entornos naturales y urbanos, y también el diseño y desarrollo de un nuevo lidar multi-haz giratorio.

En particular, esta disertación ofrece las siguientes contribuciones principales:

- **Una técnica de vecindario basada en vóxel para la clasificación de características espaciales con métodos de aprendizaje supervisado.** Mejorar la eficiencia de la clasificación de características de forma espacial a partir de datos lidar 3D es muy relevante ya que se utiliza mayoritariamente como paso fundamental previo a los retos de entendimiento de escena de alto nivel de vehículos autónomos y robots terrestres. En este sentido, el cálculo del vecindario para puntos en escáneres densos llega a ser un proceso costoso tanto para el entrenamiento como para la clasificación. Esta tesis contribuye al cálculo de vecindario basado en vóxel donde los puntos en cada vóxel no solapado en una cuadrícula uniforme son asignados a la misma clase considerando características sobre la región de soporte definida por el mismo vóxel. Por lo tanto, la nube de puntos clasificada puede ser representada por el mapa de vóxeles caracterizado geoméricamente resultante. Además, la tesis propone un nuevo marco de trabajo general para implementar y comparar diferentes clasificadores de aprendizaje supervisado. Estas contribuciones incluyen los procedimientos de entrenamiento *offline* y clasificación *online* a la vez que cinco alternativas de vector de características basado en PCA para formas tubular, plana y difusa. Además, la viabilidad de este enfoque se ha evaluado implementando una red neuronal (NN), en particular un modelo basado en perceptrón multi-capas, y también otros tres clasificadores de aprendizaje supervisado encontrados en métodos de procesamiento de escena: máquina de vector soporte (SVM), procesos Gaussianos (GP), y modelos de mezcla Gaussiana (GMM). Esta solución basada en vóxel reduce significativamente el procesamiento de datos en comparación con las técnicas basadas en punto y, por tanto, puede ser compatible con aplicaciones en tiempo real.



- **Segmentación y análisis de navegabilidad basada en mapas de vóxeles geoméricamente caracterizados.** La estimación de navegabilidad es útil para la planificación de trayectorias libres de obstáculos en tareas de navegación autónoma en entornos 3D. Basándose en GFVM, la metodología propuesta se realiza en dos fases: primero, un método de segmentación del suelo especialmente diseñado para entornos naturales, segundo, el posterior aislamiento de objetos individuales. Además, el método de segmentación del suelo está integrado en una nueva técnica de mapa de navegabilidad basado en una cuadrícula de ocupación que puede ser apropiada para robots móviles en entornos naturales. El suelo no estructurado se extrae, o sea, cada vóxel se identifica como suelo o no suelo considerando inclinaciones y superficies rugosas típicas de escenarios desestructurados. Después, la escena es dividida en objetos y además se realiza una segunda segmentación en regiones dentro de cada objeto basada en la clase geométrica del vóxel. Finalmente, el mapa de ocupación 2D propuesto representa áreas accesibles teniendo en cuenta los atributos y la altura de cada vóxel respecto al nivel del suelo en relación a las dimensiones del vehículo.
- **Diseño y desarrollo de un lidar sensor 3D portátil y asequible.** Los lidar multi-haz Velodyne están siendo cada vez más usados en muchas aplicaciones de escáner láser 3D. Sin embargo, las nubes de puntos que generan los escáneres multi-haz tienen una densidad dispersa y heterogénea, que además es aún más dispersa para el sensor VLP-16 (Puck), que es menos caro y más ligero. Esta limitación se suple parcialmente con aplicaciones móviles considerando el movimiento longitudinal en escáneres sucesivos, pero se mantiene el problema en aplicaciones estáticas o de tipo *para y continúa (stop and go)*. Esta tesis propone añadir una rotación de cabeceo al Velodyne Puck con el objetivo de incrementar la densidad, lo cual la hace más homogénea, a la vez que el campo de visión vertical. A pesar de que la agregación de un grado de libertad extra es relativamente común para escáneres 2D, esta idea no ha sido explotada con dispositivos 3D multi-haz ligeros. En concreto, se presenta un diseño compacto y ligero que puede usarse como sensor portátil. Este diseño se ha implementado en el nuevo sensor 3D *Velomotion-16*.

La producción científica de esta tesis se lista a continuación, empezando por lo más reciente:

- [66] Sensors. Analysis of 3D Scan Measurement Distribution with Application to a Multi-Beam Lidar on a Rotating Platform. 2018. (**Journal Q1**).

- [88] Sensors. Voxel-Based Neighborhood for Spatial Shape Pattern Classification of Lidar Point Clouds with Supervised Learning. 2017. (**Journal Q1**).
- [15] HRI'16, IEEE International Conference on Human-Robot Interaction. NavCue: Context immersive navigation assistance for blind travelers. 2016.
- [87] MED'15, 23rd Mediterranean Conference on Control and Automation. Occupancy grids generation based on Geometric-Featured Voxel maps. 2015.
- [85] IWANN'15, 13th International Work-Conference on Artificial Neural Networks. Multi-layer Perceptrons for Voxel-based Classification of Point Clouds from Natural Environments. 2015. (Core B)
- [84] ICIT'15, IEEE International Conference on Industrial Technology. 3D Segmentation Method for Natural Environments based on a Geometric-Featured Voxel Map. 2015.
- [86] Trabajo Fin del Máster en Mecatrónica, Universidad de Málaga. Local Semantic Map based on 3D data for navigability. 2014.
- [83] Eurathlon'14, Workshop and Summer School euRathlon Arcas. Local Semantic Map based on 3D data for navigability. 2014.
- [98] ROBOT'13, First Iberian Robot Conference. CUADRIGA: Robot Móvil para Búsqueda de Víctimas en Situaciones de Emergencia. 2013.

## Contexto

Esta tesis ha sido desarrollada en la línea de investigación de Robótica y Sistemas de Control Inteligentes del Programa de Doctorado en Ingeniería Mecatrónica de la Universidad de Málaga (UMA), enmarcado en el Programa Oficial de Doctorado del Gobierno Español (Número de registro RUCT 5600225).

Esta tesis doctoral ha sido financiada por la beca doctoral (BES-2012-061907) otorgada por el Ministerio de Economía y Competitividad de España (MINECO) asociada al Proyecto Nacional RAMBLER: Hacia la Autonomía de Robots de Exploración de Largo Alcance en Entornos Naturales (DPI2011-22443). Este proyecto pretende desarrollar e implementar un conjunto de nuevas herramientas que contribuyan al objetivo de los robots de exploración de largo alcance. Estas herramientas incluyen la generación de mapas del entorno con diferentes niveles de interpretación, tales como cartografía, ocupación, inclinaciones, clasificación de

trayectorias posibles, y también la inclusión de información semántica para la interpretación de elementos diferentes al suelo. También se abordan nuevos mecanismos 3D robustos de localización simultánea y mapeado para permitir operaciones de larga duración bajo limitada supervisión en entornos naturales.

Además, dos estancias de investigación también se financiaron por MINECO en el ámbito de las ayudas de estancias de investigación para estudiantes de doctorado. La autora pasó tres meses de estancia en el grupo IRA2 (Interaction, Réalité Augmentée, Robotique Ambiante), laboratorio de IBISC (Informatique, Biologie Intégrative et Systèmes Complexes) de la Universidad de Évry Val d'Essone en Francia continuando con la actividad de su proyecto de tesis bajo la supervisión del Dr. Fakh-Eddine Ababsa. La segunda estancia de investigación, con cuatro meses de duración, fue en el Instituto de Robótica de Carnegie Mellon University en Pittsburgh (Pennsylvania, Estados Unidos) bajo la supervisión del Dr. M. Bernardine Dias y Dr. Aaron Steinfeld. Durante esta estancia la autora participó en el proyecto *Assistive Robot for Blind Travelers*, que se describe más adelante en la lista de todos los proyectos relacionados con esta tesis. En el transcurso de estas estancias, la autora produjo como resultados las publicaciones [84] y [15], respectivamente, mencionados en la Sección Contribuciones.

La autora ha desarrollado su principal investigación en el Laboratorio de Robótica y Mecatrónica del Departamento de Ingeniería de Sistemas y Automática de la UMA. Los intereses de investigación del grupo incluyen la implementación de sistemas robóticos para intervención de emergencia en situaciones de catástrofe con el objetivo de mejorar el tiempo de respuesta y la seguridad del equipo de rescate. Algunos de los principales logros del grupo son el diseño e implementación de robots móviles que han asistido con éxito a equipos de rescate (Alacrane: [65], Quadriga: [98]). Además, el grupo centra su investigación en el desarrollo de sensores de bajo coste para capturar entornos 3D en tiempo real (Unolaser [63] y Unomotion [59]) y el reconocimiento preciso de terreno para navegación autónoma [56]. Por ello, esta disertación PhD es un paso adelante en el reconocimiento de terreno natural que puede aplicarse a la navegación robótica autónoma tal como misiones de búsqueda y rescate.

Además, la autora ha participado en otros proyectos de investigación relacionados con la tesis:

- *ATICA: Vehículo todoterreno inteligente compacto y autónomo* (806/56.3879). Financiado por CDTI y el Grupo ITURRI S.A. desde 02/07/2012 al 31/12/2014. Desarrollo de un vehículo autónomo de fuera de camino para las aplicaciones de transporte en entornos exteriores. El vehículo contiene abordo sensores (láser, GPS, IMU) y un sistema de control basado en varios procesadores con Ubuntu y la plataforma conocida ROS.

Mi contribución es la integración de la información del escáner láser 3D con el sistema de navegación mediante la generación de un mapa de ocupación con identificación de obstáculos dependiendo de su altura.

- *Navegación autónoma de un robot móvil 4x4 en terrenos naturales basado en GPS diferencial y escáner láser 3D (P10-TEP-6101-R)*. Financiado por Proyecto de Excelencia de la Junta de Andalucía desde 27/03/2013 al 27/03/2017. Mi contribución fue el desarrollo de un algoritmo para el robot Quadriga integrado con el sistema de control de movimiento en Labview [98]. El algoritmo pretende localizar víctimas mediante: *i*) un algoritmo de navegación autónoma de cobertura completa en un área determinada, integrado con Google Maps, y *ii*) seguimiento de señales Bluetooth de teléfono de víctimas.
- *First-Rob: sistema multi-robot para la cooperación de equipos de rescate de primera intervención humano y canino en escenarios de catástrofe (DPI2015-65186-R)*. Financiado por MINECO desde 01/01/2016 a 31/12/2018. La misión del proyecto es el desarrollo de un exhaustivo sistema multi-robot mecatrónico centrado en el usuario para cooperar con equipos de primera intervención con el objetivo final de reducir los riesgos para los trabajadores y mejorar su eficiencia mediante la recolección y determinación de información relevante en zonas de desastre bajo condiciones de difícil visibilidad antes de la decisión de enviar trabajadores de rescate. Este proyecto amplía el proyecto Rambler asociado a la tesis doctoral. Mi contribución es el desarrollo de un nuevo MBL.
- *Robot de asistencia para viajeros ciegos (National Robotics Initiative, IIS-1317989)*. Financiado por National Science Foundation en Carnegie Mellon University. Durante mi estancia en CMU desarrollé un algoritmo de reconocimiento facial usando ROS y OpenCV (clasificador cascade) para el robot humanoide Baxter con objeto de personalizar la interacción de Baxter con cada usuario. La mayor contribución es el cambio de canal de comunicación (voz, pantalla o gestos) del usuario multimodal dependiendo de su minusvalía sensorial (ciego, sordo, mudo) o sus preferencias. El programa conlleva la creación y actualización de una base de datos con usuarios (cara y canal de comunicación preferente) en tiempo real durante la interacción. Además desarrollé un saludo personalizado con el nombre del usuario y el apretón de manos (la localización de la mano del usuario reconociza mediante visión). También colaboré en el subproyecto NavCue, un módulo inteligente para proporcionar información basada en contexto rica y multisensorial para viajeros ciegos o con baja capacidad visual. El robot gesticula la ruta (algoritmo de planificación de ruta A\*) acompañada por

indicaciones por voz que incluyen puntos de referencia sensoriales y pistas que aportan características redundantes y memorizables para una navegación más segura.

Durante el período de formación de mi tesis he asistido a los siguientes cursos:

- Máster en Mecatrónica en la Universidad de Málaga (Enero 2013 - Julio 2014).
- Summer School en Mecatrónica, Universidad de Brno (Septiembre - Octubre 2013).
- Curso avanzado de CUDA para GPUs de Nvidia, Universidad de Málaga (Diciembre 2013 - Enero 2014).
- Sistemas Mecatrónicos - Seminario de introducción, Universidad de Málaga (Junio 2014).
- Workshop y Summer School en Robótica de euRathlon/ARCAS (Junio 2014).
- Workshop de Estructuras de Datos para Procesamiento de Nubes de Puntos 3D de Gran Escala. El 13º Congreso Internacional de Sistemas Autónomos Inteligentes, IAS13 (Julio 2014).

## Estructura de la tesis

Esta tesis se divide en seis capítulos, un apéndice y referencias bibliográficas. Excepto por este capítulo y el relacionado con las conclusiones y trabajos futuros, cada capítulo empieza con una introducción que expone el problema abordado y termina con un resumen que enfatiza las contribuciones y/o los resultados que han sido obtenidos.

El capítulo 2, *Related work*, ofrece un estado actualizado de las soluciones robóticas aplicadas a los temas o tareas implicados en esta tesis. Hay una revisión de las técnicas más comunes de representación 3D, métodos de clasificación, segmentación, mapas de ocupación, y sensores lidar.

El capítulo 3, *General framework for supervised learning classification of Lidar point clouds based on voxel-based neighborhood*, describe la solución propuesta usando diferentes clasificadores y parametrización, y analiza los resultados y rendimiento obtenidos en experimentos.

El capítulo 4, *Segmentation and navigability analysis based on geometrically featured voxel maps*, describe algunas aplicaciones para el método de clasificación de nubes de puntos 3D propuesto en el capítulo 3. Los algoritmos de segmentación y mapa de ocupación se describen y se prueban en escenarios naturales.

El capítulo 5, *Analysis and construction of a multi-beam lidar with an additional degree of freedom*, describe un nuevo escáner láser 3D de bajo coste y analiza los complejos patrones de escáner producidos por la configuración de este sensor.

El capítulo 6, *Conclusion and future work*, enfatiza las contribuciones más relevantes de esta tesis y propone temas de investigación futuros.

Finalmente, el apéndice ofrece un análisis más detallado de la arquitectura del software integrado en la plataforma ROS.

## Conclusiones

El entendimiento inteligente de escenas es un proceso complejo que se usa en muchas aplicaciones como la navegación autónoma. La información 3D precisa y de alta resolución sobre la escena puede ser obtenida como una nube de puntos 3D desde sensores como cámaras y escáneres 3D. Esta gran cantidad de datos en bruto, debe ser transformada en una representación eficiente para ayudar a procesamientos posteriores tales como la detección de objetos o el control de navegación. Esta tesis ha ofrecido contribuciones al entendimiento inteligente de escenas en entornos desestructurados basados en nubes de puntos 3D, en particular nuevos algoritmos para clasificación de nubes de puntos, segmentación de objetos, estimación de navegabilidad, y el desarrollo de un MBL rotatorio.

La clasificación de características de forma espacial basada en PCA es un componente básico en métodos recientes de procesamiento de escenas extensamente usados [49][55]. En este sentido, las características geométricas pueden describir la forma de regiones y objetos para su posterior clasificación contextual [127]. Por ello, esta tesis se ha centrado en la mejora de eficiencia, tanto en carga computacional como en precisión, de clasificación de aprendizaje supervisado de características de forma espacial (i.e. forma tubular, planar o difusa) obtenidas del análisis de covarianza [47]. Mientras que muchos autores definen categorías como la orientación vertical u horizontal [32], en este trabajo las características geométricas han sido identificadas con independencia de su orientación, lo cual amplía su aplicabilidad a entornos desestructurados.

Ya que el cálculo de vecindario para puntos 3D es un proceso costoso para la clasificación, esta tesis también ha propuesto un concepto nuevo de un marco general para implementar y comparar distintos clasificadores de aprendizaje supervisado con un cálculo de vecindario basado en vóxel. De este modo, en base a una cuadrícula 3D uniforme, las características son calculadas por cada vóxel no solapado, y todos los puntos en el mismo vóxel son asignados a la misma clase, resultando un GFVM. La viabilidad de este enfoque ha sido evaluado mediante la implementación de un método NN, en particular un modelo basado en perceptrón

multicapa, y otros tres clasificadores de aprendizaje supervisado encontrados en métodos de procesamiento de escenas (SVM, GP, y GMM) a la vez que cinco definiciones alternativas del vector de características basado en PCA para formas difusa, tubular y planar. Las métricas de rendimiento de clasificación y las medidas de tiempo de procesamiento han confirmado los beneficios del clasificador NN y la viabilidad de la técnica de vecindario basado en vóxel para escenas de lidar terrestre, también con respecto al vecindario basado en punto. Los resultados también han mostrado que la elección de las características apropiadas pueden tener un efecto impresionante en el rendimiento de las técnicas de clasificación. El método ha sido aplicado con éxito a escáneres reales tanto en entornos urbanos como naturales desde dos escáneres 3D diferentes.

Esta tesis también ha propuesto aplicaciones basadas en GFVM para la caracterización de una escena 3D. En particular, un método de segmentación y de generación de mapa de ocupación, especialmente diseñados para entornos naturales. Ambos algoritmos están basados en un método de segmentación del suelo que también amplía la representación de GFVM mediante la agregación de un nuevo atributo a los vóxeles, además de su clase geométrica. El algoritmo propuesto de segmentación del suelo tiene en cuenta inclinaciones, áreas no planas y áreas aisladas, las cuales aparecen comunmente en escenas desestructuradas. Del mismo modo, no corresponden en este caso las asunciones de suelo plano de métodos para escenas urbanas, como el ajuste de plano [125] o los umbrales de altura [53]. Para identificar vóxeles de suelo, el algoritmo consiste en un conjunto de reglas de refinamiento cuya secuencia óptima ha sido determinada mediante análisis experimental. Una vez que el suelo es segmentado, se ejecuta un método de doble segmentación primero aislando objetos, y después dividiéndolos en regiones con la misma clase geométrica. El algoritmo propuesto ha sido probado en entornos natural y urbano presentando mejores resultados en comparación con otros métodos de la literatura en entornos naturales. Además, un experimento se ha ejecutado para concluir qué tamaño de vóxel aporta el mejor resultado. Además, una métrica nueva ha sido introducida para cuantificar la precisión en la segmentación mediante una comparación entre cada nube de puntos segmentada y la misma manualmente etiquetada. Además, la autora ha propuesto un método para la generación de un mapa de ocupación 2D que es eficiente para navegación local en entornos naturales. El mapa de ocupación 2D propuesto representa áreas accesibles considerando los atributos y la altura de cada vóxel sobre el nivel del suelo en relación con las dimensiones del vehículo. Como resultado, se obtiene un mapa eficiente donde los objetos en la escenas son clasificados en clases geométricas y finalmente etiquetados como área navegable o no navegable.

Las metodologías propuestas en este trabajo pueden ser integradas en un robot móvil con un TLS o cualquier otra tecnología de captura de nubes de puntos densas 3D (vea la



Sección 2.1). Para sistemas semi-autónomos, la información geométrica y semántica puede ayudar al teleoperador a planear la ruta, como en misiones SAR donde el operador no puede acceder al área de búsqueda. Los GFVM pueden contribuir a la exploración planetaria donde el robot trata de capturar y mapear una gran cantidad de información en una estructura de datos optimizada para ahorrar memoria y ancho de banda. Además, para vehículos totalmente autónomos, el mapa de ocupación es una entrada adecuada para el algoritmo de planificación de caminos. Además, todos los descriptores deducidos de la escena 3D pueden ser procesados por un algoritmo de reconocimiento para identificar los objetos, por ejemplo, en un sistema de guiado para personas con discapacidad visual. Los problemas de registro de capturas consecutivas y mapeo se usan extensamente, también en escenarios urbanos, con el algoritmo ICP que es un proceso basado en puntos y por tanto costoso, así pues GFVM puede contribuir a reducir el coste computacional. Por ello, un algoritmo optimizado de SLAM puede producirse en base a GFVM. Además, estos métodos pueden integrarse parcialmente en vehículos aéreos, como drones, con escáneres lidar empotrados, para describir la vegetación en los bosques o detectar construcciones fraudulentas. Por lo tanto, aunque el propósito inicial de esta tesis fue contribuir a la exploración de largo alcance en entornos naturales para robots móviles terrestres, este trabajo es adecuado para una extensa variedad de aplicaciones, de hecho, el trabajo también sirve para escenas urbanas, de interior, o aéreas.

Esta tesis también ha abordado el diseño, desarrollo y análisis de un nuevo escáner láser 3D basado en sensor comercial multi-haz. En concreto, la tesis ha propuesto una rotación de cabeceo al Velodyne Puck con el propósito de incrementar la densidad, lo cual la hace más homogénea, a la vez que el campo de visión vertical. Las nubes de puntos generadas por los escáneres multi-haz tienen una densidad dispersa y heterogénea que afecta a los resultados de las técnicas de procesamiento típicas [111]. Este trabajo ha presentado un diseño compacto, asequible y ligero que puede ser usado como sensor portátil. Este diseño ha sido implementado como el nuevo sensor 3D *Velomotion-16* que mejora las limitaciones de escáneres multi-haz. Además, este trabajo ha propuesto una metodología para evaluar patrones de escaneo 3D, que ha sido aplicado para investigar la distribución de escaneo.

Resumiendo, esta tesis ha presentado contribuciones hacia el entendimiento inteligente de escenas en entornos desestructurados basadas en nubes de puntos 3D. Las principales contribuciones incluyen nuevas metodologías para la clasificación geométrica de nube de puntos, segmentación de objetos, estimación de navegabilidad y el desarrollo y análisis cualitativo de un MBL rotatorio. Ya que los datos 3D implican grandes estructuras de datos, y por tanto un alto coste computacional, la solución propuesta reduce considerablemente el coste computacional con respecto a otros métodos de mapeo 3D mediante la reducción



de la cantidad de datos a procesar. Esta tesis también ha presentado una solución de bajo coste, basada en un nuevo y asequible escáner 3D desarrollado que mejora la densidad y la homogeneidad de la nube de puntos resultante. Todas las soluciones propuestas han sido evaluadas satisfactoriamente con escáneres reales en escenarios urbanos y naturales generados por dos modelos diferentes de escáner láser. Estos métodos pueden ser aplicados a robots semi-autónomos en entornos naturales o en un escenario SAR.

## Trabajos futuros

El trabajo presentado en esta tesis podría ser ampliado con las siguientes contribuciones:

- **Integración del método completo en un vehículo autónomo**

La integración de todos los métodos propuestos, empezando por la captura de una nube de puntos 3D con el nuevo dispositivo de distancia láser, pasando por la voxelización, clasificación geométrica, segmentación y generación de mapa de ocupación para ir un paso más allá hacia la planificación de caminos y navegación autónoma. ROS ofrece implementaciones de algoritmos de planificación de caminos que pueden funcionar sobre el mapa de ocupación generado del entorno. Este trabajo de tesis está planificado para ser integrado en el robot Rambler (desarrollado en el mismo proyecto al que se asocia esta tesis).

- **Versión paralela del método de clasificación propuesto con ordenador multi procesador o GPU**

Una ventaja adicional del procesamiento sobre celdas no solapadas usando puntos que sólo pertenecen a la misma celda es que favorece la paralelización [94]. Se propone el desarrollo de una versión paralela del método propuesto para mejorar el tiempo de clasificación *online* con ordenadores multi-procesador o GPU.

- **Actualización incremental de clasificación con escáneres consecutivos**

Además del mapeado incremental con escáneres consecutivos, también puede ser interesante adaptar el método para la actualización incremental de los resultados de clasificación en lugar de procesar otra vez vóxeles repetidos. La actualización puede incluir la voxelización de nuevos puntos, la extracción de características geométricas y la clasificación de los nuevos vóxeles.

- **Clasificación de objetos en la escena**

Ya que el paso de segmentación ya divide la escena en objetos y regiones geométricas dentro de cada objeto, algunas características geométricas y estadísticas pueden ser extraídas tanto de los objetos como de las regiones para poder clasificar objetos automáticamente. Se han realizado trabajos preliminares obteniendo resultados prometedores en la identificación de árboles, coches, farolas, edificios, peatones y arbustos.

- **Mejora de Velomotion-16 desde prototipo a producto terminado**

El nuevo escáner láser es un prototipo que carece de algunas mejoras que podrían implementarse produciendo un producto final. En concreto, una interfaz de usuario para la configuración, una interfaz para el entorno ROS, optimización mecánica y la integración de baterías eficientes y compactas.

# References

- [1] Alismail, H. and Browning, B. (2015). Automatic calibration of spinning actuated lidar internal parameters. *Journal of Field Robotics*, 32(5):723–747.
- [2] Amari, S., Murata, N., Müller, K.-R., Finke, M., and Yang, H. (1996). Statistical theory of overtraining - Is cross-validation asymptotically effective? *Adv Neural Inform Process Syst*, 8:176–182.
- [3] An, S.-Y., Lee, L.-K., and Oh, S.-Y. (2015). Line segment-based fast 3d plane extraction using nodding 2d laser rangefinder. *Robotica*, 33(8):1751–1774.
- [4] Balado, J., Diaz-Vilariño, L., Arias, P., and Gonzalez-Jorge, H. (2018). Automatic classification of urban ground elements from mobile laser scanning data. *Automation in Construction*, 86:226–239.
- [5] Ball, D., Upcroft, B., Wyeth, G., Corke, P., English, A., Ross, P., Patten, T., Fitch, R., Sukkarieh, S., and Bate, A. (2016). Vision-based obstacle detection and navigation for an agricultural robot. *Journal of Field Robotics*.
- [6] Batavia, P. H., Roth, S. A., and Singh, S. (2002). Autonomous coverage operations in semi-structured outdoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 743–749 vol.1.
- [7] Baumgardner, J. R. and Frederickson, P. O. (1985). Icosahedral discretization of the two-sphere. *SIAM Journal on Numerical Analysis*, 22(6):1107–1115.
- [8] Behley, J., Steinhage, V., and Cremers, A. B. (2012). Performance of histogram descriptors for the classification of 3D laser range data in urban environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4391–4398.
- [9] Benedek, C. (2014). 3D people surveillance on range data sequences of a rotating lidar. *Pattern Recognition Letters*, 50:149–158.
- [10] Bertini, F. (2017). How to change the laser angle and FoV on a Velodyne VLP-16. <http://velodynelidar.com/blog/how-to-change-laser-angle-and-fov-vlp-16/>.
- [11] Bilmes, J. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report 510, International Computer Science Institute.
- [12] Borcs, A., Jozsa, O., and Benedek, C. (2013). Object extraction in urban environments from large-scale dynamic point cloud datasets. In *International Workshop on Content-Based Multimedia Indexing*, pages 191–194.

- [13] Brede, B., Lau, A., Bartholomeus, H. M., and Kooistra, L. (2017). Comparing RIEGL RiCOPTER UAV LiDAR derived canopy height and DBH with terrestrial LiDAR. *Sensors*, 17(10).
- [14] Cátedra de Seguridad Emergencias y Catastrofes de la Universidad de Málaga (2014, Accessed on February 2015). VIII Jornadas sobre Seguridad, Emergencias y Catástrofes.
- [15] Chen, K., Plaza-Leiva, V., Min, B., Steinfeld, A., and Dias, M. B. (2016). Navcue: Context immersive navigation assistance for blind travelers. In *IEEE International Conference on Human-Robot Interaction (HRI)*, page 559.
- [16] Chen, T., Dai, B., Wang, R., and Liu, D. (2014). Gaussian-process-based real-time ground segmentation for autonomous land vehicles. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 76(3-4):563–582.
- [17] Choe, Y., Shim, I., and Chung, M. (2011). Geometric-featured voxel maps for 3D mapping in urban environments. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 110–115.
- [18] Choe, Y., Shim, I., and Chung, M. J. (2013). Urban structure classification using the 3D normal distribution transform for practical robot applications. *Advanced Robotics*, 27(5):351–371.
- [19] Choi, C., Trevor, A., and Christensen, H. (2013). RGB-D edge detection and edge-based registration. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1568–1575.
- [20] Chromy, A. and Zalud, L. (2014). Robotic 3d scanner as an alternative to standard modalities of medical imaging. *SpringerPlus*, 3(1).
- [21] Cole, D. M. and Newman, P. M. (2006). Using laser range data for 3D SLAM in outdoor environments. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2006, pages 1556–1563.
- [22] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.
- [23] Dang, V., Kubo, M., Sato, H., Shirakawa, T., and Namatame, A. (2015). Occupancy grid map of semi-static objects by mobile observer. *Artificial Life and Robotics*, 20(1):7–12.
- [24] Dias, P., Matos, M., and Santos, V. (2006). 3d reconstruction of real world scenes using a low-cost 3d range scanner. *Computer-Aided Civil and Infrastructure Engineering*, 21(7):486–497.
- [25] Douillard, B., Brooks, A., and Ramos, F. (2009). A 3D laser and vision based classifier. In *5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 295–300.
- [26] Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A. (2011). On the segmentation of 3D LIDAR point clouds. In *IEEE International Conference on Robotics and Automation*, pages 2798–2805.

- [27] Droeschel, D., Schwarz, M., and Behnke, S. (2017). Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner. *Robotics and Autonomous Systems*, 88:104–115.
- [28] Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.
- [29] Elfarargy, M., Rizq, A., and Rashwan, M. (2013). 3D surface reconstruction using polynomial texture mapping. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8033 LNCS(PART 1):353–362.
- [30] Elhabiby, M. and Teskey, W. J. (2010). Stop-and-go 3D laser scanning and mobile mapping. *B. Sc. Thesis*.
- [31] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181.
- [32] Flynn, T., Hadjiliadis, O., and Stamos, I. (2015). Online classification in 3D urban datasets based on hierarchical detection. In *Proceedings of the International Conference on 3D Vision*, pages 380–388.
- [33] Foundation, O. S. R. (2014). Wiki ros:documentation and tutorials @ONLINE.
- [34] Gerbaud, T., Polotski, V., and Cohen, P. (2004). Simultaneous exploration and 3D mapping of unstructured environments. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 6, pages 5333–5337.
- [35] Golovinskiy, A., Kim, V., and Funkhouser, T. (2009). Shape-based recognition of 3D point clouds in urban environments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2154–2161.
- [36] Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry*, 28(5-6):367–374.
- [37] Habermann, D., Hata, A., Wolf, D., and Osorio, F. (2013). Artificial neural nets object recognition for 3D point clouds. In *Brazilian Conference on Intelligent Systems*, pages 101–106.
- [38] Hao, W. and Wang, Y. (2014). Classification-based scene modeling for urban point clouds. *Optical Engineering*, 53(3):1–9.
- [39] Himmelsbach, M., Hundelshausen, F. V., and Wuensche, H. J. (2010). Fast segmentation of 3D point clouds for ground vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 560–565.
- [40] Häselich, M., Arends, M., Wojke, N., Neuhaus, F., and Paulus, D. (2013). Probabilistic terrain classification in unstructured environments. *Robotics and Autonomous Systems*, 61(10):1051 – 1059. Selected Papers from the 5th European Conference on Mobile Robots (ECMR 2011).

- [41] Huang, W., Gong, X., and Xiang, Z. (2014). Road scene segmentation via fusing camera and lidar data. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1008–1013.
- [42] Jurman, G., Riccadonna, S., and Furlanello, C. (2012). A comparison of MCC and CEN error measures in multi-class prediction. *PLoS ONE*, 7(8).
- [43] Kaarta, Inc. (2017). Stencil data sheet. In <http://www.kaarta.com/stencil/>, accessed on October 2017, Pittsburgh, USA.
- [44] Klamt, T. and Behnke, S. (2017). Anytime hybrid driving-stepping locomotion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–8.
- [45] Kuthirummal, S., Das, A., and Samarasekera, S. (2011). A graph traversal based algorithm for obstacle detection using lidar or stereo. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3874–3880.
- [46] Lafarge, F. and Mallet, C. (2012). Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation. *International Journal of Computer Vision*, 99(1):69–85.
- [47] Lalonde, J., Vandapel, N., Huber, D., and Hebert, M. (2006). Natural terrain classification using three-dimensional lidar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839–861.
- [48] Lang, D., Friedmann, S., and Paulus, D. (2016). Adaptivity of conditional random field based outdoor point cloud classification. *Pattern Recognition and Image Analysis*, 26(2):309–315.
- [49] Lehtomäki, M., Jaakkola, A., Hyypä, J., Lampinen, J., Kaartinen, H., Kukko, A., Puttonen, E., and Hyypä, H. (2016). Object classification and recognition from mobile laser scanning point clouds in a road environment. *IEEE Transactions on Geoscience and Remote Sensing*, 54(2):1226–1239.
- [50] Leingartner, M., Maurer, J., Ferrein, A., and Steinbauer, G. (2016). Evaluation of sensors and mapping approaches for disasters in tunnels. *J. Field Robot.*, 33(8):1037–1057.
- [51] Lim, E. H. and Suter, D. (2009). 3D terrestrial LIDAR classifications with super-voxels and multi-scale conditional random fields. *CAD Computer Aided Design*, 41(10):701–710.
- [52] Linhui, L., Mingheng, Z., Lie, G., and Yibing, Z. (2009). Stereo vision based obstacle avoidance path-planning for cross-country intelligent vehicle. In *International Conference on Fuzzy Systems and Knowledge Discovery*, volume 5, pages 463–467.
- [53] Liu, H., He, G., Yu, H., Zhuang, Y., and Wang, W. (2016). Fast 3D scene segmentation and classification with sequential 2D laser scanning data in urban environments. In *Chinese Control Conference, CCC*, volume 2016-August, pages 7131–7136.
- [54] Liu, J., Liang, H., Wang, Z., and Chen, X. (2015). A framework for applying point clouds grabbed by multi-beam LIDAR in perceiving the driving environment. *Sensors*, 15(9):21931–21956.





- [55] Maligo, A. and Lacroix, S. (2016). Classification of outdoor 3D lidar data based on unsupervised gaussian mixture models. *IEEE Transactions on Automation Science and Engineering*.
- [56] Mandow, A., Cantador, T., García-Cerezo, A., Reina, A., Martínez, J., and Morales, J. (2011). Fuzzy modeling of natural terrain elevation from a 3D scanner point cloud. In *IEEE International Symposium on Intelligent Signal Processing (WISP)*, pages 171–175.
- [57] Mandow, A., Martínez, J., Reina, A., and Morales, J. (2010). Fast range-independent spherical subsampling of 3D laser scanner points and data reduction performance evaluation for scene registration. *Pattern Recognition Letters*, 31(11):1239–1250.
- [58] Martínez, J. L., Reina, A. J., Mandow, A., and Morales, J. (2012). 3d registration of laser range scenes by coincidence of coarse binary cubes. *Machine Vision and Applications*, 23(5):857–867.
- [59] Martínez, J. L., Morales, J., Reina, A. J., Mandow, A., Pequeño-Boyer, A., and García-Cerezo, A. (2015). Construction and calibration of a low-cost 3d laser scanner with 360° field of view for mobile robots. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 149–154.
- [60] Menna, M., Gianni, M., Ferri, F., and Pirri, F. (2014). Real-time autonomous 3d navigation for tracked vehicles in rescue environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 696–702.
- [61] Montemerlo, M. and Thrun, S. (2006). Large-scale robotic 3-d mapping of urban structures. In Ang, Marcelo H., J. and Khatib, O., editors, *Experimental Robotics IX*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 141–150. Springer Berlin Heidelberg.
- [62] Moon, Y. G., Go, S. J., Yu, K. H., and Lee, M. C. (2015). Development of 3D laser range finder system for object recognition. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1402–1405.
- [63] Morales, J., Martínez, J., Mandow, A., Pequeño-Boyer, A., and García-Cerezo, A. (2011). Design and development of a fast and precise low-cost 3D laser rangefinder. In *IEEE International Conference on Mechatronics*, pages 621–626.
- [64] Morales, J., Martínez, J. L., Mandow, A., Reina, A. J., Pequeño Boyer, A., and García-Cerezo, A. (2014). Bore-sight calibration of construction misalignments for 3D scanners built with a 2D laser rangefinder rotating on its optical center. *Sensors*, 14(11):20025–20040.
- [65] Morales, J., Martínez, J. L., Mandow, A., Seron, J., and García-Cerezo, A. J. (2013). Static tip-over stability analysis for a robotic vehicle with a single-axle trailer on slopes based on altered supporting polygons. *IEEE/ASME Transactions on Mechatronics*, 18(2):697–705.
- [66] Morales, J., Plaza-Leiva, V., Mandow, A., Gomez-Ruiz, J. A., Serón, J., and García-Cerezo, A. (2018). Analysis of 3d scan measurement distribution with application to a multi-beam lidar on a rotating platform. *Sensors*, 18(2):395.

- [67] Mountrakis, G., Im, J., and Ogole, C. (2011). Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(3):247–259.
- [68] Mutz, F., Veronese, L., Oliveira-Santos, T., De Aguiar, E., Auat Cheein, F., and Ferreira De Souza, A. (2016). Large-scale mapping in complex field scenarios using an autonomous car. *Expert Systems with Applications*, 46:439–462.
- [69] Na, H. J., Choe, Y., and Chung, M. J. (2014). Efficient 3D terrain mapping based on normal distribution transform grid. In *International Conference on Control, Automation and Systems*, pages 656–660.
- [70] Na, K., Byun, J., Roh, M., and Seo, B. (2013). The ground segmentation of 3D LIDAR point cloud with the optimized region merging. In *International Conference on Connected Vehicles and Expo (ICCVE)*, pages 445–450.
- [71] Neumann, T., Dülberg, E., Schiffer, S., and Ferrein, A. (2016). A rotating platform for swift acquisition of dense 3D point clouds. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9834:257–268.
- [72] Neumann, T., Ferrein, A., Kallweit, S., and Scholl, I. (2014). Towards a mobile mapping robot for underground mines. In *Proceedings of the 7th IEEE Robotics and Mechatronics Conference (RobMech-14)*.
- [73] Nguyen, D. V., Kuhnert, L., Jiang, T., Thamke, S., and Kuhnert, K. D. (2011). Vegetation detection for outdoor automobile guidance. In *Proceedings of the IEEE International Conference on Industrial Technology*, pages 358–364.
- [74] Omar, T. and Nehdi, M. L. (2016). Data acquisition technologies for construction progress tracking. *Automation in Construction*, 70:143–155.
- [75] Othmani, A., Piboule, A., Dalmau, O., Lomenie, N., Mokrani, S., and Voon, L. (2014). Tree species classification based on 3D bark texture analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8333:279–289.
- [76] Ozbay, B., Kuzucu, E., Gul, M., Ozturk, D., Tasci, M., Arisoy, A. M., Sirin, H. O., and Uyanik, I. (2015). A high frequency 3d lidar with enhanced measurement density via papoulis-gerchberg. In *International Conference on Advanced Robotics, ICAR 2015*, pages 543–548.
- [77] Pan, R. and Taubin, G. (2016). Automatic segmentation of point clouds from multi-view reconstruction using graph-cut. *Visual Computer*, 32(5):601–609.
- [78] Papadakis, P. (2013). Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373 – 1385.
- [79] Papon, J., Abramov, A., Schoeler, M., and Worgotter, F. (2013). Voxel cloud connectivity segmentation - supervoxels for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2027–2034.



- [80] Patterson, D. W. (1996). *Artificial Neural Networks: Theory and Applications*. Prentice-Hall Series in Advanced Communications. Prentice Hall.
- [81] Paulsen, R., Bærentzen, J., and Larsen, R. (2010). Markov random field surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):636–646.
- [82] Pfrunder, A., Borges, P. V. K., Romero, A. R., Catt, G., and Elfes, A. (2017). Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D lidar. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2601–2608.
- [83] Plaza, V. and A., M. (2014). Local semantic map based on 3D data for navigability. In *Workshop and Summer School euRathlon Arcas*.
- [84] Plaza, V., Ababsa, F., García-Cerezo, A., and Gomez-Ruiz, J. A. (2015a). 3D segmentation method for natural environments based on a geometric-featured voxel map. In *IEEE International Conference on Industrial Technology (ICIT)*, pages 1602–1607.
- [85] Plaza, V., Gomez-Ruiz, J. A., Mandow, A., and García-Cerezo, A. J. (2015b). Multi-layer perceptrons for voxel-based classification of point clouds from natural environments. In *13th International Work-Conference on Artificial Neural Networks (IWANN)*, pages 250–261.
- [86] Plaza-Leiva, V. (2014). Local semantic map based on 3D data for navigability. Master's thesis, Univeristy of Málaga, ETSII.
- [87] Plaza-Leiva, V., Gomez-Ruiz, J., Ababsa, F., Mandow, A., Morales, J., and García-Cerezo, A. (2015). Occupancy grids generation based on geometric-featured voxel maps. In *Mediterranean Conference on Control and Automation (MED)*, pages 766–771.
- [88] Plaza-Leiva, V., Gomez-Ruiz, J., Mandow, A., and García-Cerezo, A. (2017). Voxel-based neighborhood for spatial shape pattern classification of lidar point clouds with supervised learning. *Sensors*, 17(3):594.
- [89] Rabbani, T., van den Heuvel, F. A., and Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. In *IEVM06*.
- [90] Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.
- [91] Reina, A., Martínez, J., Mandow, A., Morales, J., and García-Cerezo, A. (2014). Collapsible cubes: Removing overhangs from 3D point clouds to build local navigable elevation maps. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1012–1017.
- [92] Rekleitis, I., Bedwani, J.-L., and Dupuis, E. (2009). Autonomous planetary exploration using LIDAR data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3025–3030.
- [93] Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

- [94] Rusu, R. B., Sundareshan, A., Morisset, B., Hauser, K., Agrawal, M., Latombe, J. C., and Beetz, M. (2009). Leaving flatland: Efficient real-time three-dimensional perception and motion planning. *Journal of Field Robotics*, 26(10):841–862.
- [95] Santamaria-Navarro, A., Teniente, E. H., Morta, M., and Andrade-Cetto, J. (2015). Terrain classification in complex three-dimensional outdoor environments. *Journal of Field Robotics*, 32(1):42–60.
- [96] Schubert, S., Neubert, P., and Protzel, P. (2016). How to build and customize a high-resolution 3D laserscanner using off-the-shelf components. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9716:314–326.
- [97] Seo, B. and Chung, M. (2013). Traversable ground detection based on geometric-featured voxel map. In *Korea-Japan Joint Workshop on Frontiers of Computer Vision*, pages 31–35.
- [98] Serón, J. and García-Cerezo, J.A. and Plaza, V. and Martínez J.L. (2013). Cuadriga: Robot móvil para búsqueda de víctimas en situaciones de emergencia. In *First Iberian Robot Conference (ROBOT)*.
- [99] Shaukat, A., Blacker, P. C., Spiteri, C., and Gao, Y. (2016). Towards camera-LIDAR fusion-based terrain modelling for planetary surfaces: Review and analysis. *Sensors*, 16(11).
- [100] Sheh, R., Jamali, N., Kadous, M. W., and Sammut, C. (2006). A low-cost, compact, lightweight 3D range sensor. In *Australasian Conference on Robotics and Automation, ACRA 2006*.
- [101] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.
- [102] Sima, M.-C. and Nüchter, A. (2013). An extension of the Felzenszwalb-Huttenlocher segmentation to 3D point clouds. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 8783, pages 878302–878302–6.
- [103] Souza, A. and Maia, R. (2013). Occupancy-elevation grid mapping from stereo vision. In *Robotics Symposium and Competition (LARS/LARC), 2013 Latin American*, pages 49–54.
- [104] Sun, S. and Salvaggio, C. (2013). Aerial 3D building detection and modeling from airborne lidar point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1440–1449.
- [105] Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition*. Academic Press, 4th edition.
- [106] Ueda, T., Kawata, H., Tomizawa, T., and Ohya, A. (2006). Mobile sokuiki sensor system-accurate range data mapping system with sensor motion. In *International Conference on Autonomous Robots and Agents*.

- [107] Velodyne LiDAR, I. (2012). *HDL-32E, User's Manual and Programming Guide*. Velodyne LiDAR, Inc.
- [108] Velodyne LIDAR, Inc. (2017a). Datasheets. In <http://velodynelidar.com/docs/datasheet/>, accessed on November 2017, Morgan Hill, CA, USA.
- [109] Velodyne LIDAR, Inc. (2017b). Products. In <http://velodynelidar.com/products.html>, accessed on November 2017, Morgan Hill, CA, USA.
- [110] Vitali, A. and Rizzi, C. (2017). A virtual environment to emulate tailor's work. *Computer-Aided Design and Applications*, 14(5):671–679.
- [111] Vlaminck, M., Luong, H., Goeman, W., and Philips, W. (2016). 3D scene reconstruction using omnidirectional vision and lidar: A hybrid approach. *Sensors*, 16(11).
- [112] Vu, H., Nguyen, H. T., Chu, P. M., Zhang, W., Cho, S., Park, Y. W., and Cho, K. (2017). Adaptive ground segmentation method for real-time mobile robot control. *International Journal of Advanced Robotic Systems*, 14(6).
- [113] Weingarten, J. and Siegwart, R. (2006). 3d slam using planar segments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3062–3067.
- [114] Wulf, I. and Wagner, B. (2003). Fast 3D Scanning Methods for Laser Measurement Systems. In *Proceedings of the International Conference on Control Systems and Computer Science*, volume 1, pages 312–317, Bucharest, Romania.
- [115] Xiao, J., Zhang, J., Adler, B., Zhang, H., and Zhang, J. (2013). Three-dimensional point cloud plane segmentation in both structured and unstructured environments. *Robotics and Autonomous Systems*, 61(12):1641 – 1652.
- [116] Xiong, X., Munoz, D., Bagnell, J. A., and Hebert, M. (2011). 3-D scene analysis via sequenced predictions over points and regions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2609–2616.
- [117] Xu, Z., Wei, J., and Chen, X. (2015). Vehicle recognition and classification method based on laser scanning point cloud data. In *Proceedings of the International Conference on Transportation Information and Safety*, pages 44–49.
- [118] Yandun, F., Reina, G., Torres-Torriti, M., Kantor, G., and Auat Cheein, F. (2017). A survey of ranging and imaging techniques for precision agriculture phenotyping. *IEEE/ASME Transactions on Mechatronics*.
- [119] Yang, B., Dong, Z., Zhao, G., and Dai, W. (2015). Hierarchical extraction of urban objects from mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 99(Supplement C):45 – 57.
- [120] Yebes, J. J., Bergasa, L. M., and García-Garrido, M. (2015). Visual object recognition with 3d-aware features in kitti urban scenes. *Sensors*, 15(4):9228–9250.

- [121] Yoshida, T., Irie, K., Koyanagi, E., and Tomono, M. (2010). A sensor platform for outdoor navigation using gyro-assisted odometry and roundly-swinging 3d laser scanner. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1414–1420.
- [122] Yu, Y., Li, J., Guan, H., Wang, C., and Yu, J. (2015). Semiautomated extraction of street light poles from mobile LIDAR point-clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 53(3):1374–1386.
- [123] Zhang, F. and Knoll, A. (2016). Vehicle detection based on probability hypothesis density filter. *Sensors*, 16(4).
- [124] Zhou, Q.-Y. and Neumann, U. (2013). Complete residential urban area reconstruction from dense aerial LiDAR point clouds. *Graphical Models*, 75(3):118–125.
- [125] Zhou, Y., Yu, Y., Lu, G., and Du, S. (2012). Super-segments based classification of 3D urban street scenes. *International Journal of Advanced Robotic Systems*, 9(248):1–8.
- [126] Zhu, X., Zhao, H., Liu, Y., Zhao, Y., and Zha, H. (2010). Segmentation and classification of range image from an intelligent vehicle in urban environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1457–1462.
- [127] Zhuang, Y., Liu, Y., He, G., and Wang, W. (2015). Contextual classification of 3D laser points with conditional random fields in urban environments. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 3908–3913.

# Appendix A

## 3D data processing

### A.1 Robotic operating system (ROS)

ROS [33] is a framework which involves tools and drivers for programming robots, it is open source and BSD licensed. The main idea of ROS is the communication between modules from a robot, in one or more processors, in an easy and flexible way. The programming of these modules is made in C++ or Python, and ROS can be installed over several operative systems but only is considered stable in Ubuntu.

The main advantage of ROS is the distributed and modular design. It is based on a structure divided in processes, called nodes, which relate among themselves. This way, the processing takes place on the different nodes, which exchange information through messages. These nodes may be designed individually, even written in different programming languages, and be used in a flexible way on real time. The communication between the nodes it can be made by a message passing mechanism, as publisher-subscriber method, or by request of services from a client, as client-server approach. The basic units from the ROS architecture are described below:

- Node: it is the smallest structure in a ROS network, it can be considered as an executable with an specific purpose within a robot control system. Nodes communicate to each other through message passing. Executing a node requires the command `roslaunch` whose syntax is: `roslaunch package_name node_name _parameter_name:=value`. Notice the name of each parameter should be preceded by the character `_` as prefix. Executing ROS code requires previously the initialization of the master node with the command `roslaunch`.
- Master node, also called `roslaunch`: it is the central server of the architecture, responsible for keeping a register of all the nodes from the platform, and keeping track of all the

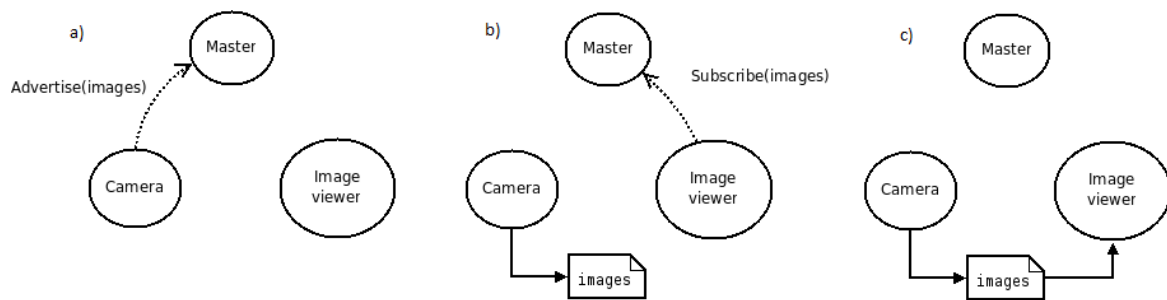


Fig. A.1 Topic based communication in ROS

messages that can be exchanged by the nodes. Thus, this node let other two different nodes find each other when they want to share information. This two nodes can be in different computers and just need to know where is the master node. Once they have been in contact, they communicate peer-to-peer.

- **Message:** nodes communicate to each other through message passing. A message is a data structure which involves different fields depending on the information to send. Messages are transmitted by two different ways: through topics or through services.
- **Topic:** it is one of the two options to the message exchange in ROS. Topics are like paths which messages can be exchanges through. A topic is always associated to an specific message type. When a node want to publish a message, it indicates the message type to the master node and register it with a topic name. Secondly, if another node want to receive that message, it will subscribe to the some topic by indicating it is ready to the master node. Then the master node will establish the communication between the two nodes, publisher and subscriber, and the information exchange will be made in a peer-to-peer transmission (see Fig A.1). Hence, a topic is a name used for identify an specific message exchange. To check if the communication is success, there are useful commands such as 'rostopic list -v' to show all the active publisher and subscriber nodes, and 'rostopic echo topicname' to show the data published on a specified topic.
- **Service:** it is the second way offered by ROS for the message exchange. The communication request-response is made using services which are defined by a pair of messages, one for the request and other for the response, both defined as `service_type`. This concept uses the client-server model. As in the Topic based communication, previously to the peer-to-peer communication, both nodes has to register in the master node, which is responsible to make them being in contact (see Fig A.2). The main difference with the topic based communication is that in this way the emitter only publish a message

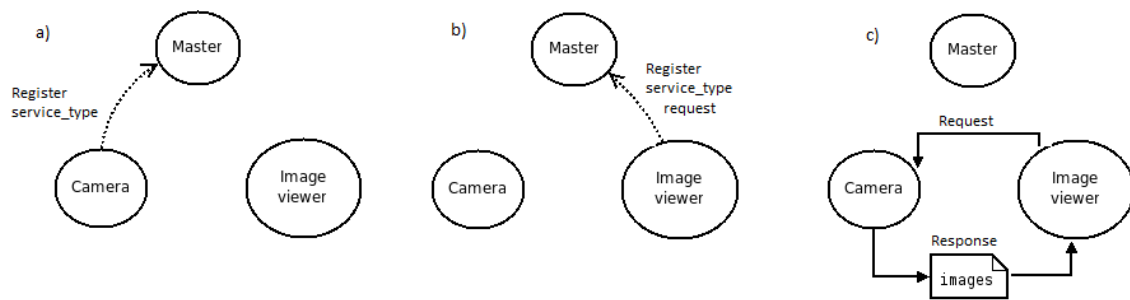


Fig. A.2 Service based communication in ROS

when a receiver requests it. On the contrary, in the previous approach the messages were published continuously, it is a faster protocol, but if there is no subscriber it involves a waste of resources.

- **Nodelet:** it is an executable code designed to provide a way to run multiple algorithms on a single machine, in a single process, without incurring copy costs when passing messages intraprocess. The communication is optimized to do zero copy pointer passing between publish and subscribe calls within the same node. In fact, it offers better performance when the nodes are running in the same machine instead of a network or distributed system. Executing a nodelet it is the same as a node by using `roslaunch`, but it does not require any prefix in the `parameter_name`.
- **Launchfile:** it is an XML file which is executable. It calls one or more nodes or nodelets with the specified parameters for being executed in the same or different machines. It is specially useful in large projects which require several nodes execution, each one with several parameters. To execute one of them the command is: `roslaunch package_name launchfile_name parameter_name:=value`.

In the programming level, ROS is based on a filesystem organized in a hierarchy of blocks (see Fig. A.3). Packages are the main unit in ROS organization. Each package provides a different service and involves different nodes which, as previously shown, are executable processes. ROS is composed by hundreds of software packages for the robotic system development. The packages with the same subject/device are grouped in a stack. Furthermore, all the different stacks from the same user or development group are packed in a repository.



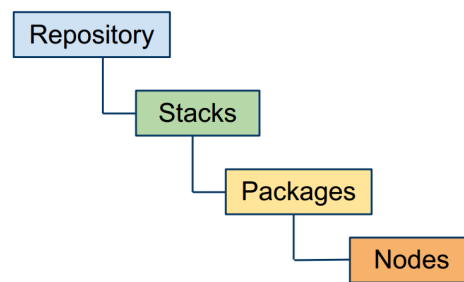


Fig. A.3 ROS code hierarchy

VERSION	PCD format version
FIELDS	Field names of points
SIZE	Size of each field, in bytes
TYPE	Type of each field: I for char, short or int, U for unsigned char/short/int, F for float
COUNT	Elements of each field, by default 1, for instance the x field has one element
WIDTH	Point Cloud width, in number of points
HEIGHT	Point Cloud height, in number of points: 1 if is a unorganized point cloud.
VIEWPOINT	View point for the acquisition, in translation (tx ty tz) + quaternion (qw qx qy qz)
POINTS	Total number of points.
DATA	Codification for the points, ASCII or binary.

Table A.1 PCD header structure

## Point Cloud Library in ROS

PCL is a large scale, open project [93] for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them.

PCL is released under the terms of the 3-clause BSD license and is open source software. It is free for commercial and research use. PCL is cross-platform, and has been successfully compiled and deployed on Linux, MacOS, Windows, and Android/iOS. In addition, it is fully integrated with ROS.

PCL uses the PCD file format (Point Cloud Data) in order to describe the n-dimensional point clouds. This file has a header coded in ASCII (see Tab. A.1) and the points information can be in ASCII or binary. An example of this format can be seen in Fig. A.4. As it shows, it is the seventh version of PCD format, each point has fields for the Cartesian coordinates xyz



```

# .PCD v.7 - Point Cloud Data file
format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 3
HEIGHT 3
VIEWPOINT 0 0 0 1 0 0 0
POINTS 9
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
0.97192 0.278 0 4.2108e+06
0.944 0.29474 0 4.2108e+06
0.98111 0.24247 0 4.2108e+06
0.93655 0.26143 0 4.2108e+06
0.91631 0.27442 0 4.2108e+06
0.81921 0.29315 0 4.2108e+06

```

Fig. A.4 Example of a PCD file

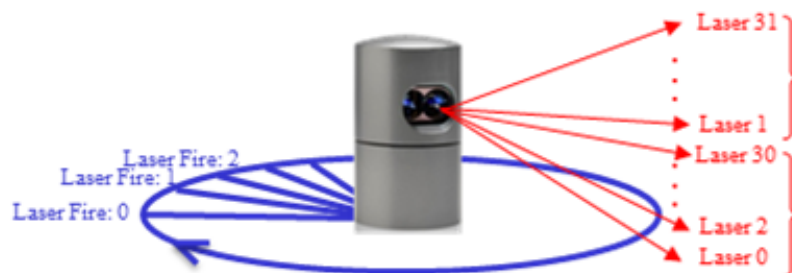


Fig. A.5 Velodyne HDL-32E [107]

and also a color RGB field, data are 4bytes float type and they are coded in ASCII, and the size of the cloud is 3x3 points.

## A.2 Velodyne HDL-32E

Velodyne HDL-32E [107] is a laser rangefinder, also called LIDAR (Laser Imaging Detection and Ranging). It consists in a technology to measure the distance from an emitter and an object or surface in front of it by using a pulsed laser beam. This model provides 32 vertically aligned lasers <sup>1</sup> in a platform which rotates around a vertical axis in clockwise direction.

<sup>1</sup> Other Velodyne models provide 16 or 64 laser emitters.

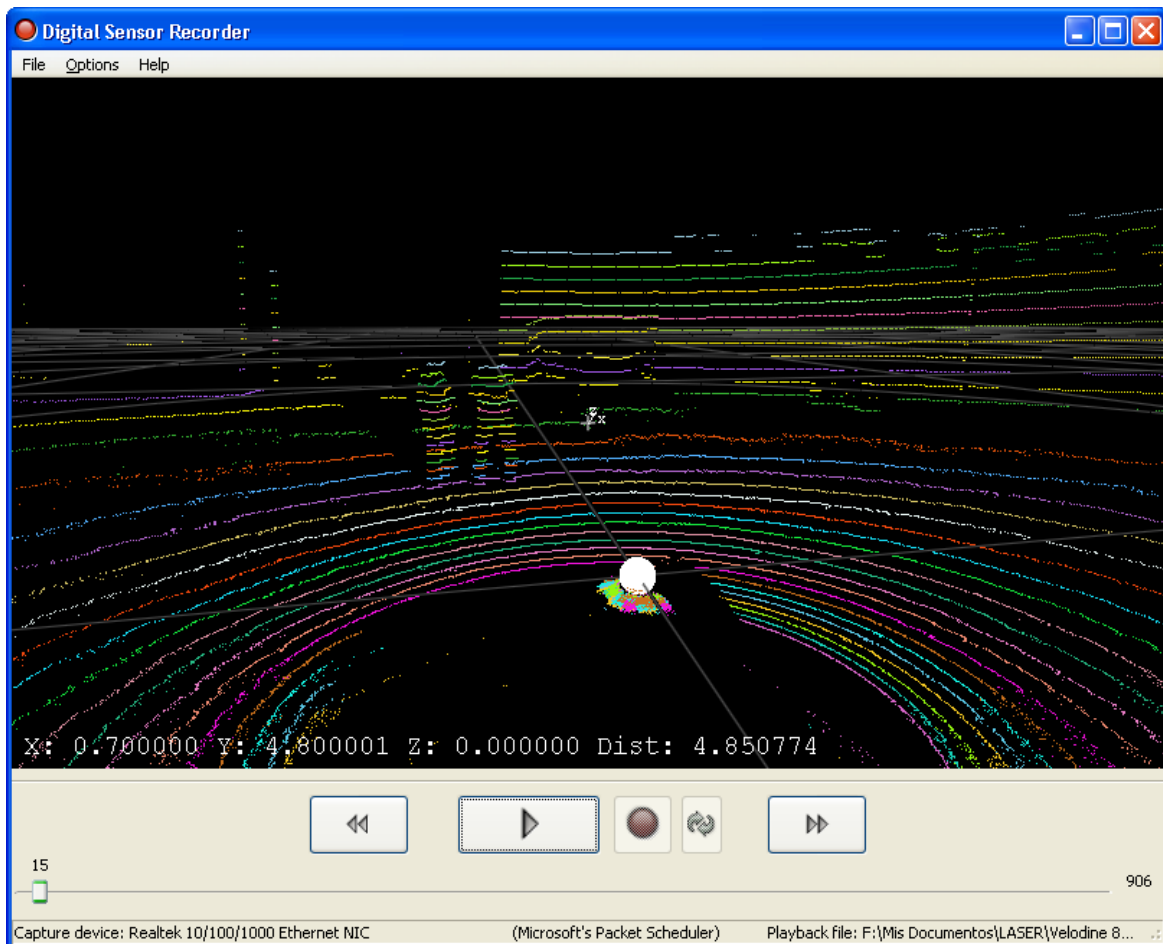


Fig. A.6 Velodyne capture shown in *DSR viewer*

This way the device generate a 3D point cloud of a 360° scene (see Fig. A.5). The device offers a 41.3° vertical field of view, usable returns up to 70 meters with an accuracy of  $\pm 2\text{cm}$ , and approximately 700 thousand laser shots per second (depending on the revolution speed which can be configured, more speed less points).

The lasers emitters are divided in two blocks, 16 lasers each one. The first block is the lower one and it includes the even lasers from 0 to 30 sorted from the lower to the upper one, and the second block includes the odd lasers from 1 to 31 sorted in the same way. The header of the device rotates horizontally at the same time it is firing the lasers. During the fire time the distance from all the lasers are measured by order from 0 to 31, it means from the bottom to the top alternating both blocks of lasers.

In addition to the power adaptor, the device is connected through a Ethernet cable with a computer. Velodyne send the data through Ethernet to the broadcast address of a local net with IP 192.168.3.0. This means, in order to use Velodyne firstly it is necessary to set up the

```

8 0.003869 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
9 0.004424 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
10 0.004975 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
11 0.005530 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
12 0.006081 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
13 0.006635 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
14 0.006964 192.168.18.33 255.255.255.255 UDP Source port: 8308 Destination port: 8308
15 0.007189 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
16 0.007740 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
17 0.008295 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable
18 0.008846 192.168.18.33 255.255.255.255 UDP Source port: opentable Destination port: opentable

Frame 1 (1248 bytes on wire, 1248 bytes captured)
Ethernet II, Src: 60:76:88:20:12:21 (60:76:88:20:12:21), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 192.168.18.33 (192.168.18.33), Dst: 255.255.255.255 (255.255.255.255)
User Datagram Protocol, Src Port: opentable (2368), Dst Port: opentable (2368)
Data (1206 bytes)

000 ff ff ff ff ff ff 60 76 88 20 12 21 08 00 45 00 .....V...!.E.
010 04 d2 00 00 40 00 ff 11 a4 51 c0 a8 12 21 ff ff .....@...Q...|..
020 ff ff 09 40 09 40 04 be 00 00 ff ee 2c 27 a6 07 ...@.@...|...
030 c3 4b 17 c3 f6 07 c3 32 17 87 3c 08 af 14 17 d7 ..K.....2...<..
040 aa 08 af 19 17 af 0f 09 c3 08 17 c3 7b 09 af f6 .....|...
050 16 9b 08 0a c3 08 17 9b 95 0a c3 00 00 ff 42 0b .....B...
060 c3 00 00 ff 08 0c c3 00 00 ff d8 0c d7 00 00 ff .....
070 05 0e ff 00 00 ff b1 0f eb 00 00 ff e1 11 eb 00 .....
080 00 ff a4 14 ff 00 00 ff 49 17 c3 00 00 ff ff ee .....|...Q...>..
090 3b 27 a6 07 c3 10 17 d7 f5 07 c3 71 16 9b 3e 08 .....
0a0 af 71 16 eb ae 08 af 3c 16 c3 0d 09 c3 2a 16 d7 ..q.....<.....%..
0b0 78 09 af 0f 16 af 0a 0a c3 1a 16 af 95 0a c3 00 x.....
0c0 00 ff 3c 0b c3 00 00 ff 04 0c c3 00 00 ff d7 0c .....|...
0d0 d7 00 00 ff 04 0e ff 00 00 ff b3 0f eb 00 00 ff .....<.....
0e0 ef 11 d7 00 00 ff a1 14 ff 00 00 ff 72 16 d7 00 .....|...P...
0f0 00 ff ff ee 4d 27 a7 07 c3 25 16 eb f7 07 c3 16 .....M...%.....

```

Fig. A.7 PCAP file from Velodyne shown by *Wireshark*

network card of the PC which is connected to. The PC should have any valid IP within the mentioned network <sup>2</sup>. Data are returned like ethernet packets with UDP protocol and saved in a binary file with PCAP extension. This files and real time captures can be shown with the program *DSR\_viewer* in Windows XP provided by Velodyne (see Fig. A.6), or *Veloview* in Windows 7 in free distribution on internet. Colors represents each one from the 32 lasers, which means each ring in the image are the measures from the same laser emitter.

Furthermore, PCAP files can be analyzed with a network explorer, for instance *Wireshark* (see Fig. A.7) running on Linux operative system.

## Transformation from Velodyne RAW data to standard point cloud

To transform data from Velodyne in a Cartesian point cloud, a C program has been implemented. The algorithm opens a captured data file, which uses PCAP extension, and join Ethernet packages in frames. Each frame involves a point cloud with a 360° scene, then the information is converted to Cartesian coordinates of each point, and the result is saved as a text file with the list of the XYZ points.

A PCAP file is composed by a file header with a size of 24 Bytes, and a list of Ethernet packages where each one is preceded by a PCAP header of 16 Bytes (see Tab. A.2). GPS data are optional packages, only sent if Velodyne is connected to its GPS device. One PCAP file can store all the capture data during the execution of the device, hence a longer capturing process means a bigger output file. There are two kinds of Ethernet packages from Velodyne,

<sup>2</sup>To change the IP address, in Windows or Linux, you have to go through the menus Network properties→TCP/IP and change the following values to IP 192.168.3.100, Network mask 255.255.255.0 and Gateway 192.168.3.1.

File header	24 Bytes
Ethernet packet 1 header	16 Bytes
Ethernet packet 1 (distance data)	1248 Bytes
Ethernet packet 2 header	16 Bytes
Ethernet packet 2 (GPS data)	554 Bytes
...	
Ethernet packet n header	16 Bytes
Ethernet packet n (distance data)	1248 Bytes

Table A.2 PCAP file structure from Velodyne

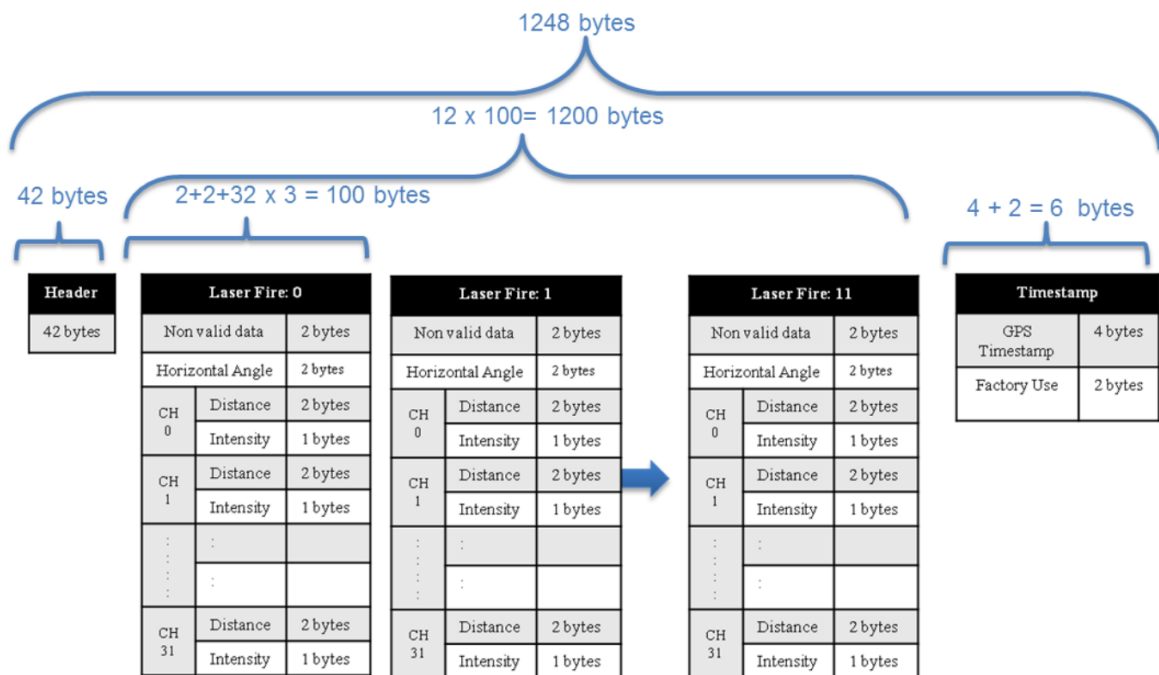


Fig. A.8 Structure of an Ethernet distance packet from Velodyne

distance data with a size of 1248 Bytes, or GPS positioning, 554 Bytes. In our experiments the GPS attached to the rangefinder will not be connected, so only distance data packages will be considered. The distance data packages (see Fig. A.8) involves a header of 42 bytes, 12 laser firings, each one employ 100 bytes, and a timestamp of 6bytes. Each laser firing includes a rotation position, this is an horizontal angle, and values from the 32 lasers, distance and intensity. All the fields from the distance data package are transmitted with the less significative byte in the first place (Little Endian).

The algorithm to convert the PCAP file in a Cartesian point cloud is described in the algorithm 1. The PCAP file involves several completed revolutions from Velodyne, however the algorithm will extract only the first frame or 360° scene. Since the size of each field

of the binary file are known, the file descriptor can be moved to the init of the firing data and read only this avoiding the headers. This way the buffer for reading the file does not need more space than 1200 bytes. Every Laser fire has its own horizontal angle, but the end of a frame could be in the middle of an ethernet package. For this reason, in each fire a function is called, which detect the end of the frame and hence the end of the algorithm. This function save the initial horizontal angle ( $\phi$ ), and detect when the complete revolution has come comparing the current horizontal angle of each fire. Then, for each one of the 32 lasers, the vertical angle ( $\theta$ ) is read from the constant input array and the distance field ( $d$ ) is composed rearranging the two separated bytes -because they were transmitted as Little Endian- and converting to the meters unit. At this point the spherical coordinates are known for the current point, so they can be transformed to Cartesian coordinates according to the equations A.1.

---

**Algorithm 1** Conversion from PCAP to Cartesian Point Cloud
 

---

**Require:** PCAP file and vertical angles array

**Ensure:** Cartesian Point Cloud in a text file

```

1: Open PCAP file
2: loop
3:   firingData  $\leftarrow$  Read next Ethernet Package
4:   for  $fire = 1$  to 12 do
5:      $\phi \leftarrow$  Read horizontal angle
6:     if Detect end of 360° scene then
7:       return
8:     end if
9:     for  $laser = 1$  to 32 do
10:       $\theta \leftarrow$  Read vertical angle
11:       $d \leftarrow$  Read and rearrange field distance
12:       $x,y,z \leftarrow$  Convert polar coordinates to Cartesian ( $d, \theta, \phi$ )
13:      Print point on output file ( $x,y,z$ )
14:    end for
15:  end for
16: end loop

```

---

$$\begin{aligned}
 x &= d \sin \theta \cos \phi \\
 y &= d \sin \theta \sin \phi \\
 z &= d \cos \theta
 \end{aligned}
 \tag{A.1}$$

## Velodyne in ROS

Alternatively, the RAW data from Velodyne can be transformed to a list of Cartesian points by using the library PCL for ROS in Linux. The difference with our program is that ROS provide fixed size messages from RAW Velodyne data, and it does not make group with the information from the same frame (a whole 360° revolution). The main advantage of ROS is the real time processing, because all the algorithms are optimized, even parallelized if the current computer is multicore.

Once ROS is successfully installed on a Ubuntu system, Velodyne package for ROS has to be installed too by using the command: `sudo apt-get install ros-VERSION-velodyne`, where `VERSION` has to be replaced by the name of the ROS version, such as `groovy` or `hydro`. To work with Velodyne package in ROS, in addition to change the PC IP (described in Section A.2), it is necessary to add a route indicating the IP of the input data device, this is the Veodyne fixed IP, in our case 192.168.18.108. In new versions of the Velodyne HDL-32E firmware, the device IP can be changed.

- **VELODYNE\_DRIVER** Captures data from the device (or from a previous capture in a .pcap file) and publishes them in RAW with `VelodyneScan.msg` message type with topic `/velodyne_packets` and `frame_id` `velodyne`. Each message contains points from a whole revolution, one frame.

```
Rosrun velodyne_driver velodyne_node _model:=32E
```

It requires run `roscore` previously. Optionally can publish a previous capture instead of data from device by using: `_pcap:=$(pwd)/path/filename.pcap`

```
Roslaunch velodyne_driver nodelet_manager.launch model:=32E
```

As the previous node but in a `nodelet` wrap. Same parameters but without the prefix `_`.

```
Rosrun velodyne_driver vdump file_prefix eth0
```

Captures RAW data from topic `/velodyne_packets` and save them in a PCAP file. The output file is called `file_prefix000`. Optionally, the net interface can be set, by default `eth0`. A single PCAP file may have data from more than one revolution from the Velodyne device, so it correspond to a video of the capture. <sup>3</sup>

- **VELODYNE\_POINTCLOUD** This feature captures RAW data from topic `/velodyne_packets` (produced by any of the `velodyne_driver` commands), transform it and publish it in a new message as an standard point cloud type: `sensor_msgs/PointCloud2.msg`

<sup>3</sup>If the device is on a moving platform, as a robot, the frames can be mapped creating a video of the route.

in topic `/velodyne_points` and `frame_id=velodyne`. Each point is described by its Cartesian coordinates  $(x,y,z)$ , the measure intensity, and a ring value. It can be launched in three different ways: in a node, in a nodelet, and in a launch file which call the nodelet shown below respectively:

```
Rosrun velodyne_pointcloud cloud_node _calibration:=/path/32db.yaml
```

```
Rosrun nodelet nodelet standalone velodyne_pointcloud/CloudNodelet
calibration:=/path/32db.yaml
```

```
Roslaunch velodyne_pointcloud cloud_nodelet.launch calibration:=
/path/32db.yaml
```

Furthermore the package provides a launch file which calls `velodyne_driver` followed by `velodyne_pointcloud`.

```
Roslaunch velodyne_pointcloud 32e_points.launch
```

It also can replay a previous capture by adding :

```
pcap:=~/velodyne/capture_filename.pcap.
```

- **POINTCLOUD\_TO\_PCD** subscribes the input topic and convert the point cloud from one message to a pcd file, and save it. One PCD file is saved for each frame o 360° scan. These output files can be processed with PCL. By default, PCD files are coded in ASCII and the point cloud data can be loaded in Matlab (and show with `verpuntos.m`) by removing manually the PCD header.

```
rosrun pcl_ros pointcloud_to_pcd input:/velodyne_points prefix:=
~/velodyne/pcd/ Where ' /velodyne/pcd/' the directory to store the output PCD
files. By default, output files are saved in /home/<username>/.ros/.
```

- **OCTOMAP** is a library for managing point clouds with an octree model. `Octomap_mapping` suscribe a pointcloud topic and publish the voxelized information. First step is the installation on ROS:

```
Sudo apt-get install ros-VERSION-octomap ros-*VERSION-octomap-
mapping ros-VERSION-octomap-rviz-plugins
```

\*ROS VERSION =groovy, hydro..

Second step is configuring the file `octomap_server octomap_mapping.launch`:

```
<param name="frame_id" type="string" value="velodyne" />
```

```
<remap from="cloud_in" to="/velodyne_points" />
```



Finally launching octomap server:

```
Roslaunch octomap_server octomap_mapping.launch
```

- RVIZ. ROS provides a powerful viewer for any kind of ROS message including point clouds. For instance, after launching 32e\_points.launch, we can run rviz in a different terminal in Ubuntu with: `Rosrun rviz rviz -f velodyne`<sup>4</sup> The rviz node subscribes the same topic and frame\_id and show the static scene or the sequence of them. The first time you use rviz, it is necessary to set up it, by doing 'Add Display → Message type: PointCloud2 → Topic: "/code>velodyne\_points"'. Following execution of rviz node will open the same configuration by default. The colours of the points correspond to the height of them (see Figure A.9). If octomap is running we can also represent the occupied voxels by adding Display MarkerArray, topic /occupied\_Cells\_vis\_array, fixed\_frame=velodyne. Colours correspond to the height of points.

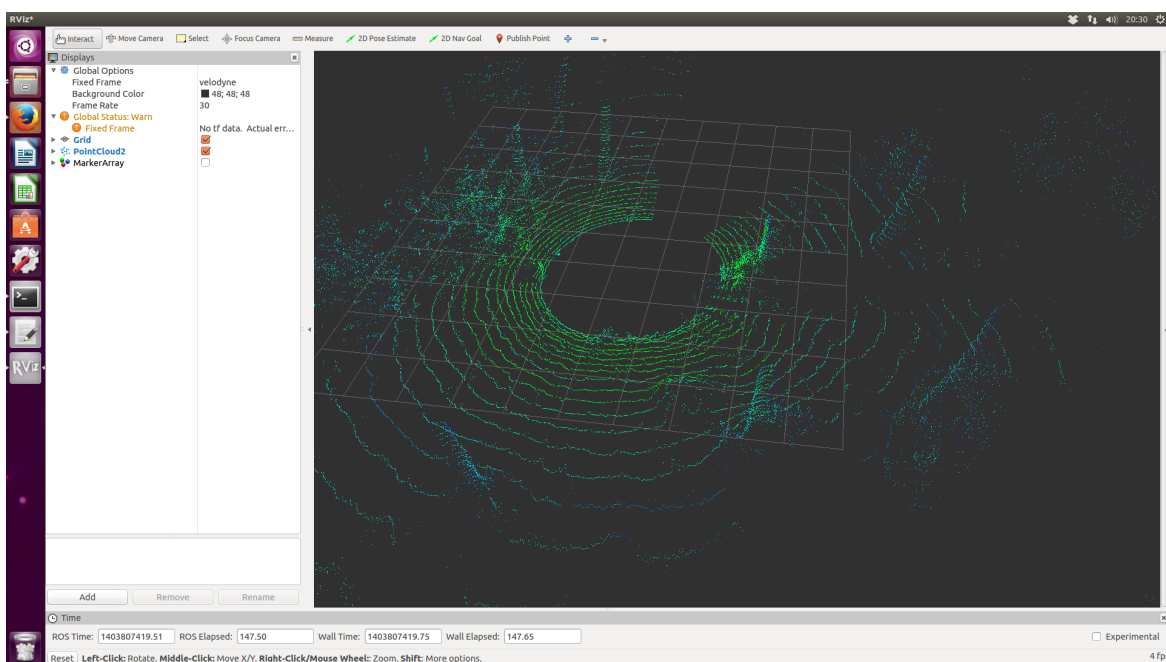


Fig. A.9 Rviz ROS viewer showing a Velodyne point cloud

<sup>4</sup>Without the parameter -f velodyne in the command the frame\_id would be configured inside rviz





Fig. A.10 Unolaser LRF

### A.3 Unolaser

UnoLaser 30M135Y [63] is a low-cost laser rangefinder based on pitching a commercial 2d rangefinder, Hokuyo UTM-30LX (see Fig. A.10). It has a maximum scanning range of 30 m and a minimum range of 0.1 m. It is powered with a DC power supply of 12 V with a nominal consumption of 14.4 W with peaks of 33.6 W. A complete 3D scan can be obtained with a maximum pitch resolution of  $0.067367^\circ$  in 95.75 s and with a minimum pitch resolution of  $4.16129^\circ$  in 1.55 s. This sensor is suitable for scanning both indoor and outdoor environments, as Velodyne. This device offers a measurement resolution 1 mm and a Field of view  $270^\circ \times 131^\circ$ . The point cloud captured by Unolaser is in a shorter range than Velodyne, but it is denser and rings do not appear in the scene. Vertical resolution in Velodyne is not high enough to capture small details from the objects in the scene (see figure...). On the contrary, the objects are well defined (see Fig. A.11). Due to the fact that unolaser has only one laser emitter, scanning a scene is slower than velodyne which has 32 or 64 laser emitters depends on the model. Furthermore, Velodyne is rotating continuously around a vertical axis, whereas unolaser has two different movements. The Hokuyo piece make the horizontal scan with a continuous rotation with a  $90^\circ$  of death angle in the back side. Unolaser adds the third dimension by adding a not continuous pitch rotation. This way the rotation is not completed and, what is more important, a big delay is added. A new model of Unolaser is being designed to improve this drawback. It will be able to turn continuously and make the scanning process faster.

The output of each scan involves two different files with the name of the timestamp and 'scan3d\_' as a prefix and extension .dat. The binary file is called in the same way adding the suffix \_raw before the extension. The text file involves many rows as horizontal scans, it depends on the quantity of vertical scans, velocity of it and the resolution parameters. Each horizontal scan has several columns. The first one indicates the vertical angle, it can be from  $-45^\circ$  to  $86^\circ$ . The second column is the timestamp, by default each horizontal scan differs from

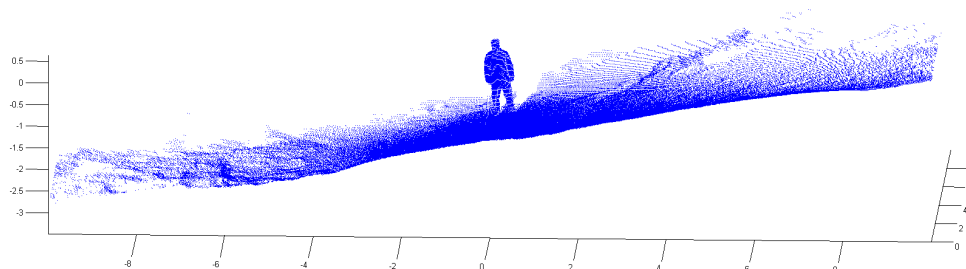


Fig. A.11 Point cloud captured by Unolaser

the previous one in 50ms. From the third until the last column are the distances measured in the horizontal scan ( $270^\circ$ ). This measure is wrong if the value is 1 or bigger than 30000mm, which is 30m.

## A.4 Velomotion

The ROS architecture implemented for Velomotion described above is shown in Figure A.12

The original files have been created in a ROS package *velomotion*<sup>5</sup> but some functionalities come from other ROS packages, as the philosophy of modularity of ROS aims.

The created *velomotion* package contains the following original source files. Some of them are launch files, which describe the nodes that should be run and parameters that should be set, in XML format, and hence are interpreted files, non compiled. And the other ones are source codes of new nodes in C, that of course have been compiled inside the new ROS package.

- *velomotion.launch*: calls to other two launch files, *positionInTf* and *VLP\_transform.launch*, as a result produces a point cloud generated by Velodyne and it is already transformed into the inclination of the velomotion engine.
  - *VLP\_transform.launch* is in *velodyne\_pointcloud* package of ROS, and it calls to *velodyne\_node* to obtain the distances and also calls to *transform\_node* to trans-

<sup>5</sup>For simplicity and comfort for the developer the source files could be compiled inside *epos\_hardware* package avoiding creating the new package *velomotion*

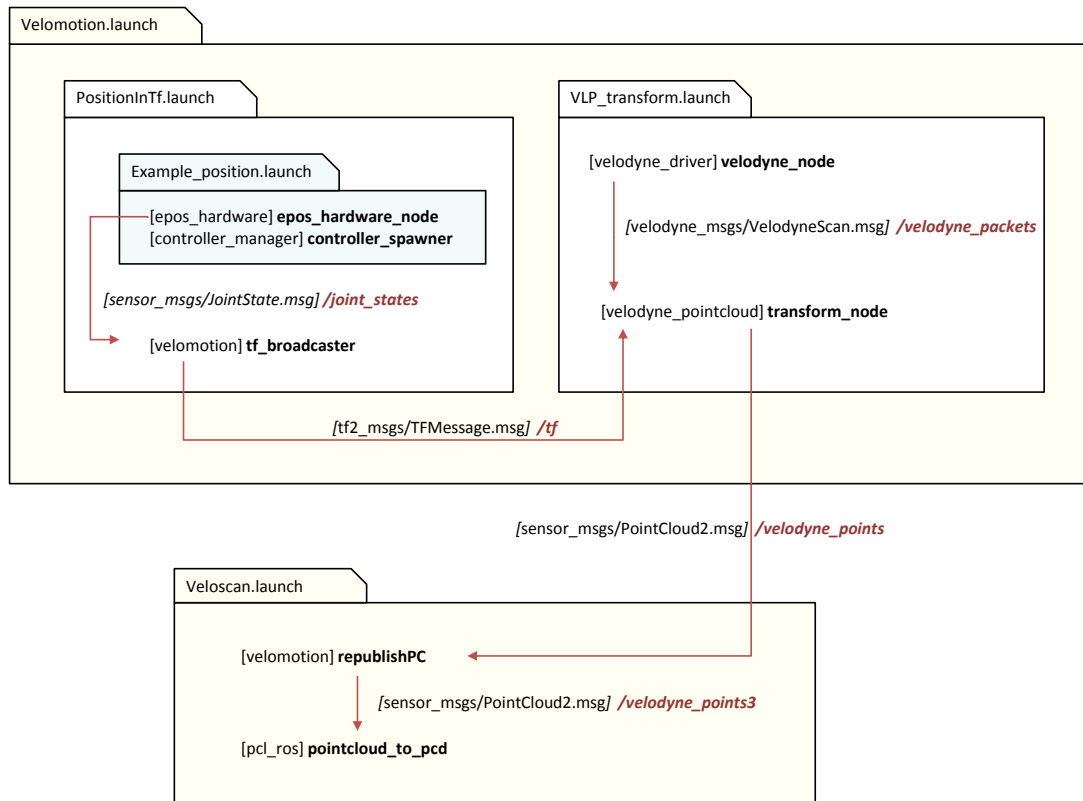


Fig. A.12 Velomotion communication with ROS.

form those points into Cartesian point cloud taking into account the inclination of the coordinate centre of the engine.

- `positionInTf.launch`: generates the coordinates transformation message using `Example_position.launch` and the original node `tf_broadcaster`.
  - `example_position.launch` is included in `epos hardware` package. `epos hardware_node` generates the message `/joint_states` with the position of the engine. The node `controller_spawner` load and start the indicated nodes at the same time.
- `tf_broadcaster.c`: read the `joint_state` and generates the position in coordinates transform format in the message `/tf`.
- `veloscan.launch`: receives the consecutives point clouds from `velodyne` and generate the accumulated point cloud in a PCD file. Contains the original node `republishPC` and `pointcloud_to_pcd` from a standard package in ROS.

- pointcloud\_to\_pcd: receives the point cloud in a message and write it in a PCD file.
- republishPC.c: it runs the movement to the velomotion engine and accumulate the point clouds during that movement. It generates the final dense point cloud when the engine get to its end position, and then it returns to its origin point.

Figure A.12 shows the hierarchy of files and the messages flux between nodes. Launch files contains calls to other ones besides calls to nodes. Nodes are processes that communicate with each other through streaming topics using a publisher/subscriber communication where the ROS Master is the core element. That means, even if the figure shows direct communication between nodes, inside the architecture, the messages are published and subscribed through the master node - at first. The figure uses the following notation:

- boxes with the name of the launch file wich contains inside calls to other launch or nodes.
- nodes are identified by its container package, in brackets, and the name of the node in bold text: [name\_of\_package] **name\_of\_node**
- messages are identified by its container package and message type, in brackets and italic, and the name of the topic for the communication, in italic and red: [*name\_of\_package/ message\_type*] / *name\_of\_topic*. And the red arrows represents the direction publisher -> subscriber nodes respectively.