UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

Automatización Avanzada y Robótica

Tesis Doctoral

3D MAPPING AND PATH PLANNING FROM RANGE DATA

Ernesto Homar Teniente Avilés

Director: Juan Andrade-Cetto

Desembre 2015

Universitat Politècnica de Catalunya
BarcelonaTech (UPC)

PhD program:
AUTOMATIZACIÓN AVANZADA Y ROBÓTICA

The work presented in this thesis has been carried out at:

Institut de Robòtica i Informàtica Industrial, CSIC-UPC

Thesis supervisor:
Juan Andrade-Cetto

Con amor y entrega para mi amada familia ...

With love and devotion to my beloved family ...

Agradecimientos

Me gustaría agradecer profundamente al Dr. Juan Andrade-Cetto por guiarme a través del doctorado, por su apoyo, paciencia y sus consejos en los aspectos, académicos y personales a lo largo del trayecto, y por su comprensión en los momentos difíciles. Sus invaluables consejos, su continua ayuda y constante motivación han dado como fruto esta tesis.

Quiero agradecer a mis amigos en México, Emmanuel Navarro, Luis Antonio, Carlos Roberto, Pamela, Carlos Souvervielle y Victor Osuna, quienes fueron testigos del iniciar de mi partir y andadura; a las grandes amistades descubiertas durantes estos años, Agustín Ortega, Omar Mendez y Cecy, Kim Quadrado, Ben Piegne y Sonia, Charly Anderle, Luis Yerman, Anaís Garrell, Michael Villamizar, Blanca Vela, Alex Pérez y Gina, y Carlos Paredes por su apoyo en lo académico y personal cuando las cosas no fueron como era esperado, por las horas de conversaciones y momentos compartidos; también agradezco a Leonel Rozo, Oscar Sandoval, Carlos Lara, Michael Hernandez, Farzad Husain y Rafael Valencia, Carlos Moya, Paco Alarcón, Jordi Esteban y Gavin Craig por su grata compañia. Gracias a mis amigos de la grupeta de ciclismo, deporte integral en mi vida, Jordi, Hita, Dani, Xavi, Julià, Claudio, Pablo y Jack, que mediante la convivencia sin que quizas lo supieran me han aportado fuerza para seguir. A mis amigos de despacho, Eduard Trulls, Gonzalo Ferrer, Aleix Rull, Marti Morta y Alex Goldhoorn. Agradezco a Saskia y familia Castro, por el tiempo compartido, apoyo y afecto. Gracias también German, Jaume, Jesús, Rubén y Fabián. De la misma manera, muchas gracias a todos los que conforman el Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Quiero agradeces a todas a aquellas personas que con cuyos consejos y comentarios han contribuido positivamente en mi vida personal y académica durante este trayecto que fue mas largo de lo esperado. Asi mismo quiero agradecer

por su apoyo al Dr. Alberto San Feliu.

Nancy gracias por estar a mi lado, por tu compañia en esas últimas largas noches de escritura, por lo que hemos compartido en el andar, que con tu apoyo y comprensión me has motivado e impulsado a trabajar aún en momentos difíciles, por lo que a día a día aportas en mi vida, te amo enormemente.

Agradezco a mi amada familia, a mis padres Ernesto y Zaray a quienes amo incodicionalmente, por su inestimable apoyo a lo largo de estos años, gracias por sus consejos en los momentos difíciles y a cualquier hora que los necesitara, gracias a mis hermanos, Michel, Jessica, Luis Fernando y Erika, por su muestras de cariño aún en la distancia, los amo. Agradezco a mi abuelos Luis, Rosalba q.e.p.d., Josefina y mi padrino Carlos, por su fé en mi. Al resto de familia muchas gracias por su inmenso e incondicional amor. ¡Gracias!.

Así mismo me gustaría reconocer todas las fuentes de apoyo financiero que de una manera u otra contribuyeron para el desarrollo de esta tésis. Las cuales incluyen: al Consejo Mexicano de Ciencia y Tecnología (CONACyT) con una beca para estudios doctorales en el extrajero; a la Universitat Politècnica de Catalunya; al proyecto europeo URUS (No. IST FP6 STREP 045062); los proyectos nacionales de investigación PAU y PAU+ (DPI2008-06022 y DPI2011-2751) y MIPRCV Consolider Ingenio (CSD2007-018),financiados por el Ministerio Español de Economía; y al grupo consolidado de Investigación VIS (SGR2009-2013).

Acknowledgements

I would like to deeply thank Dr. Juan Andrade-Cetto for guiding me through my PhD studies, for his support, patience and advice in all academic and personal issues along the way, and for his understanding in difficult times. His priceless advice, continued support and constant motivation have resulted in this thesis.

I want to thank my friends in Mexico, Emmanuel Navarro, Luis Antonio, Carlos Roberto, Pamela, Carlos Souvervielle and Victor Osuna, who witnessed my depart and the beginning of this walk; to the great discovered friendships during these years, Agustín Ortega, Omar Mendez and Cecy, Kim Quadrado, Ben Peigne and Sonia, Charly Anderle, Luis Yerman, Anaís Garrell, Michael Villamizar, Blanca Vela, Alex Pérez and Gina, and Carlos Paredes for their support in academics and personal issues when things did not go as well as it was expected, for the hours of conversations and shared moments; I also thank Leonel Rozo, Oscar Sandoval, Carlos Lara, Michael Hernandez, Farzad Husain, Rafael Valencia, Carlos Moya, Paco Alarcón, Stephen and Gavin Craig Jordi for his pleasant company. Thanks to my cycling group of friends, the sport of my life, Jordi, Hita, Dani, Xavi, Julià, Claudio, Paul and Jack, that by riding together, without knowing, they gave me strength to continue. My office friends, Eduard Trulls, Gonzalo Ferrer, Aleix Rull, Martí Morta, and Alex. Thanks Saskia and the Castro family, for the sharing time, support and affection. Thanks to German, Jaume, Jesus Ruben and Fabian. Similarly, thank you very much to all who make part of Institut de Robòtica i Informàtica Industrial (CSIC-UPC). I want to thank all those whose advice and comments contributed positively in my personal and academic life during this journey, which was longer than expected.

Nancy, thanks for being by my side, for your company in those last long writing nights, for what we have shared in the walk. With your support and understanding

you've inspired and encouraged me to work even in difficult moments, so that every day you bring in my life, I love you tremendously.

I thank my beloved family, my parents, Ernesto and Zaray whom I love unconditionally, for their invaluable support over the years, thank you for your advice in difficult times, anytime I needed it. Thanks to my sisters and brother, Michel, Jessica, Luis Fernando and Erika for their displays of affection even in the distance, I love you. I thank my grandparents Luis, Rosalba R.I.P., Josefina and my godfather Carlos, for his faith in me. To the rest of the family thank you very much for your immense and unconditional love. Thank you!.

Also I would like to acknowledge all sources of financial support that in one way or another contributed to the development of this thesis. These include the Mexican Council of Science and Technology (CONACYT) with a scholarship for PhD studies; the Technical University of Catalonia; the European project URUS (IST FP6 STREP No. 045062); the national research projects PAU and PAU + (DPI2008-06022 and DPI2011-2751) and MIPRCV Consolider Ingenio (CSD2007-018), financed by the Spanish Ministry of Economy; and the consolidated group Research VIS (SGR2009-2013).

Abstract

This thesis reports research on mapping, terrain classification and path planning. These are classical problems in robotics, typically studied independently, and here we link such problems by framing them within a common proprioceptive modality, that of three-dimensional laser range scanning. The ultimate goal is to deliver navigation paths for challenging mobile robotics scenarios. For this reason we also deliver safe traversable regions from a previously computed globally consistent map.

We first examine the problem of registering dense point clouds acquired at different instances in time. We contribute with a novel range registration mechanism for pairs of 3D range scans using point-to-point and point-to-line correspondences in a hierarchical correspondence search strategy. For the minimization we adopt a metric that takes into account not only the distance between corresponding points, but also the orientation of their relative reference frames. We also propose FaMSA, a fast technique for multi-scan point cloud alignment that takes advantage of the asserted point correspondences during sequential scan matching, using the point match history to speed up the computation of new scan matches. To properly propagate the model of the sensor noise and the scan matching, we employ first order error propagation, and to correct the error accumulation from local data alignment, we consider the probabilistic alignment of 3D point clouds using a delayed-state Extended Information Filter (EIF). In this thesis we adapt the Pose SLAM algorithm to the case of 3D range mapping, Pose SLAM is the variant of SLAM where only the robot trajectory is estimated and where sensor data is solely used to produce relative constraints between robot poses. These dense mapping techniques are tested in several scenarios acquired with our 3D sensors, producing impressively rich 3D environment models.

The computed maps are then processed to identify traversable regions and to plan

navigation sequences. In this thesis we present a pair of methods to attain high-level off-line classification of traversable areas, in which training data is acquired automatically from navigation sequences. Traversable features came from the robot footprint samples during manual robot motion, allowing us to capture terrain constrains not easy to model. Using only some of the traversed areas as positive training samples, our algorithms are tested in real scenarios to find the rest of the traversable terrain, and are compared with a naive parametric and some variants of the Support Vector Machine. Later, we contribute with a path planner that guarantees reachability at a desired robot pose with significantly lower computation time than competing alternatives. To search for the best path, our planner incrementally builds a tree using the A* algorithm, it includes a hybrid cost policy to efficiently expand the search tree, combining random sampling from the continuous space of kinematically feasible motion commands with a cost to goal metric that also takes into account the vehicle nonholonomic constraints. The planer also allows for node rewiring, and to speed up node search, our method includes heuristics that penalize node expansion near obstacles, and that limit the number of explored nodes. The method book-keeps visited cells in the configuration space, and disallows node expansion at those configurations in the first full iteration of the algorithm. We validate the proposed methods with experiments in extensive real scenarios from different very complex 3D outdoors environments, and compare it with other techniques such as the A*, RRT and RRT* algorithms.

Resumen

Esta tesis versa sobre la construcción de mapas, la clasificación del terreno y la planificación de rutas para aplicaciones de robótica móvil. Estos problemas, clásicos de la robótica, se estudian habitualmente de forma independiente, y en esta tesis, los abordamos de forma conjunta, vinculados a una modalidad sensorial concreta, la telemetría láser tridimensional. El objetivo final es ofrecer rutas de navegación para entornos complejos de robótica móvil.

Examinaremos en primer lugar el problema de cómo registrar en un sistema de referencia común, nubes de puntos densas adquiridas con un telémetro láser tridimensional. Contribuimos con un novedoso mecanismo de registro de nubes de puntos jerárquico que analiza la correspondencia punto a punto en las capas de menor resolución y la correspondencia punto a plano en capas con más nivel de detalle. Para la minimización de la función de coste, adoptamos una medida que tiene en cuenta no sólo la distancia entre puntos correspondientes, sino también la orientación de sus planos de referencia relativos. También proponemos FaMSA, una técnica rápida para el registro de múltiples nubes de puntos capturadas con un telémetro láser de forma continuada. La técnica aprovecha las correspondencias entre puntos obtenidas en iteraciones anteriores para acelerar el cómputo de nuevas hipótesis de correspondencia. Para propagar adecuadamente el modelo de ruido del sensor, empleamos un método de propagación de incertidumbre de primer orden. Y, para conseguir un registro global y coherente de todas las nubes de puntos, adaptamos una técnica de SLAM desarrollada en nuestro grupo de investigación que utiliza un filtro de información con una ventana temporal de estados (EIF). En esta variante, que se llama PoseSLAM, mantiene una estimación probabilística de la historia del robot y de un pequeño número de observaciones que permiten cerrar lazos. La construcción de mapas con PoseSLAM y con las técnicas de

registro de nubes de puntos desarrolladas en esta tesis permiten construir mapas 3D con suficiente nivel de detalle para poder planificar trayectorias de movimiento complejas para robots móviles en entornos exteriores.

Identificamos mediante técnicas de clasificación aquellas zonas del mapa que son transitables. Los métodos que se presentan con este fin en esta tesis requieren una fase de entrenamiento automático a partir por ejemplo, de las mismas secuencias de navegación que se emplearon para construir el mapa. Los métodos de clasificación propuestos se comparan de forma favorable frente a otros métodos recientes. Una vez se ha determinado el espacio libre de colisiones por el que el robot podría navegar, en esta tesis presentamos también métodos que emplean estos mapas clasificados para la planificación de trayectorias que garantice no solo la accesibilidad del robot sino una ruta óptima en distancia recorrida. Para buscar el mejor camino entre un punto de origen y otro de destino en el mapa, nuestro planificador construye incrementalmente un árbol utilizando el algoritmo A*, incluyendo además una política de coste híbrido que toma en cuenta el modelo cinemático del vehículo tanto para explorar el espacio de acciones desde el punto actual, como para estimar una cota inferior del coste de llegar a la meta. El método de exploración elimina nodos del árbol mediante la reconexión de nodos con una heurística que penaliza la expansión de nuevos nodos cerca de obstáculos. El método de exploración presentado se ha validado extensamente en bases de datos propias conseguidos con los robots del IRI y con bases de datos de terceros.

Resum

Aquesta tesi versa sobre la construcció de mapes, la classificació del terreny i la planificació de rutes per a aplicacions de robòtica mòbil. Aquests problemes, clàssics de la robòtica, s'estudien habitualment de forma independent, i en aquesta tesi, els abordem de forma conjunta, vinculats a una modalitat sensorial concreta, la telemetría làser tridimensional. L'objectiu final és oferir rutes de navegació per a entorns complexos de robòtica mòbil.

Examinarem en primer lloc el problema de com registrar en un sistema de referència comuna, núvols de punts denses adquirides amb un telémetre làser tridimensional. Contribuïm amb un nou mecanisme de registre de núvols de punts jeràrquic que analitza la correspondència punt a punt en les capes de menor resolució i la correspondència punt a plànol en capes amb més nivell de detall. Per a la minimització de la funció de cost, adoptem una mesura que té en compte no només la distància entre punts corresponents, sinó també l'orientació dels seus plànols de referència relatius. També proposem FaMSA, una tècnica ràpida per al registre de múltiples núvols de punts capturades amb un telémetre làser de forma continuada. La tècnica aprofita les correspondències entre punts obtingudes en iteracions anteriors per accelerar el còmput de noves hipòtesis de correspondència. Per propagar adequadament el model de soroll del sensor, emprem un mètode de propagació d'incertesa de primer ordre. I, per aconseguir un registre global i coherent de tots els núvols de punts, adaptem una tècnica de SLAM desenvolupada en el nostre grup de recerca que utilitza un filtre d'informació amb una finestra temporal d'estats (EIF). En aquesta variant, que es diu PoseSLAM, es manté una estimació probabilística de la història del robot i d'un petit nombre d'observacions que permeten tancar llaços. La construcció de mapes amb PoseSLAM i amb les tècniques de registre de núvols de punts desenvolupades en aquesta tesi permeten construir mapes 3D amb

suficient nivell de detall per poder planificar trajectòries de moviment complexes per a robots mòbils en entorns exteriors.

Identifiquem mitjançant tècniques de classificació aquelles zones del mapa que són transitables. Els mètodes que es presenten a aquest efecte en aquesta tesi requereixen una fase d'entrenament automàtic a partir per exemple, de les mateixes seqüències de navegació que es van emprar per construir el mapa. Els mètodes de classificació proposats es comparen de forma favorable enfront d'altres mètodes recents. Una vegada s'ha determinat l'espai lliure de col·lisions pel qual el robot podria navegar, en aquesta tesi presentem també mètodes que empren aquests mapes classificats per la planificació de trajectòries que garanteixi no solament l'accessibilitat del robot sinó una ruta òptima en distància recorreguda. Per buscar el millor camí entre un punt d'origen i un altre de destinació en el mapa, el nostre planificador construeix incrementalment un arbre utilitzant l'algorisme A*, incloent a més una política de cost híbrid que pren en compte el model cinemàtic del vehicle tant per explorar l'espai d'accions des del punt actual, com per estimar una cota inferior del cost d'arribar a la meta. El mètode d'exploració elimina nodes de l'arbre mitjançant la reconnexió de nodes amb una heurística que penalitza l'expansió de nous nodes prop d'obstacles. El mètode d'exploració presentat s'ha validat extensament en bases de dades pròpies aconseguits amb els robots de l'IRI i amb bases de dades de tercers.

Contents

List of Figures

List of Symbols

Mathematical Notation

$\bar{f}_*$	Gaussian process posterior mean
$\mathbf{f}_*$	Gaussian process (posterior) prediction (random variable)
$\hat{\mathbf{n}}$	unit normal vector with coordinates $(\hat{n}_x, \hat{n}_y, \hat{n}_z)$
$\boldsymbol{I}$	index matrix list: $I = I_{u,v,w}$
$\boldsymbol{I}$	a list of points
$\mathbf{k}_*$	vector, short for $K(X, \mathsf{x}_*)$, when there is only a single test case
$\mathbf{n}$	normal vector with coordinates (n_x, n_y, n_z)
$\mathbf{t}$	translational vector with coordinates (x,y,z)
$\mathbf{u}$	control sequence inputs $(v, \omega, \Delta t)$
$\mathbf{x}$	state vector, history of robot poses from time 0 to n :
$\mathcal{C}$	the robot configuration's space
$\mathcal{C}_{\text{free}}$	collision free configuration space
$\mathcal{C}_{\text{obs}}$	obstacles configuration space
Δ t	time increment
κ	surface curvature

$\mathcal{GP}$	Gaussian process: $f(\mathsf{x}) \sim \mathcal{GP}(m(\mathsf{x}), k(\mathsf{x}, \mathsf{x}'))$, the function f is distributed as a Gaussian process with by a mean function $m(\mathsf{x})$ and the covariance function $k(\mathsf{x}, \mathsf{x}')$
$\mathbb{V}[f_*]$	predective variance
$\mathcal{O}(\cdot)$	computational complexity
x	robot position and attitude
f	any features
f_r	feature, local surface roughness
f_s	feature, local surface slope
p	point with coordinates (x,y,z)
p_k	point with coordinates (x,y,z) at time k
x	Gaussian process input vector
y	Gaussian process output target
y_*	Gaussian process training single test point output or target
μ	mean
ω	rotational velocity
ϕ	vector of hyperparameters (parameters of the covariance function)
ρ	vehicle turning radius
Σ	covariance matrix
σ	variance
σ_f^2	variance of the noise free signal
σ_n^2	noise variance
τ	robot trajectory

υ translational velocity

$\mathbf{X}$ sates space

$\mathbf{X}_{free}$ collision free states space

C number of classes in a classification problem

I the identity matrix

K or $K(X, X)$ nxn covariance (or Gram) matrix

$k(\mathsf{x}, \mathsf{x}')$ covariance (or kernel) function evaluated at x and x'

$m(\mathsf{x})$ the mean function of a Gaussian process

P points cloud

P' query points cloud

$p(y \mid x)$ or $y \mid x$ conditional random variable y given x and its probability (density)

P_k points cloud at time k

P_o obstacle points points: $P_o \in P$

P_t traversable/navigable points: $P_t \in P$

p_t probability threshold

R rotation matrix

$R(\mathbf{n}, \theta)$ axis angle rotation matrix

t time

T_p local planar patches or tangent plane

X $D \times n$ matrix of the training inputs $\{\mathsf{x}_i\}_{i=1}^n$: the design matrix

X_* matrix of test inputs

$YorZ$ corresponding closest points between two point clouds

q node in the tree structure

Acronyms

2D Two dimensional space

3D Three dimensional space

6D Six dimensional space, normally position and attitud

ANN Approximate Nearest Neighbor

DOF Degrees-of-freedom

EIF Extended Information Filter

EKF Extended Kalman Filter

EP Expectation Propagation

GCP Grid Closes Point

GP Gaussian process

HRA* Hybrid randomized path planning

ICP Iterative Closest Point

LOO Leave-One-Out approach

LP Laplace approximation

MSE Mean Square Error

NNS Nearest Neighbor Search

POMDP Partially Observable Markov Porcesses

SLAM Simultaneous localization and mapping

Chapter 1

Introduction

A large number of mobile robot applications are designed to coexist in human environments; vacuum cleaners, floor cleaners, AGVs for warehouse distribution, line-side logistics, hospital robots, museum tour guides; they all are expected to navigate and move efficiently alongside people. Most of these mobile robot applications, whether for indoor scenarios or not, assume that the world where the robot moves is a plane [40, 47, 118, 127]. However, more challenging applications that are truly outdoor encounter uneven characteristics of the terrain, ramps, stairs, and other unpredictable elements in the environment. All these elements significantly difficult the navigation task. We are interested in planning navigation paths for these more challenging mobile robotics scenarios. To alleviate the localization problem and to help the robot take appropriate navigation decisions, we are interested in building full 3D maps for our robots, and to do it as autonomously as possible. In this context, our robot must be able to build a locally detailed and globally consistent map. Once the map is built some post-processing might be necessary to obtain a complete digital compact volumetric model, merging information to reduce the memory space of the map without compromising details and consistency. The robot must then be able to perform a traversability analysis in order to take the appropriate navigation decisions.

The proprioceptive modalities that can be used for mapping vary from: ultrasonics devices, typically used for indoor mapping [30, 85], to 2D and 3D range data used for indoor and outdoor mapping, respectively [40, 47, 54, 148]; to vision, which is a cheaper sensing technique with the ability to use appearance information during data association [66, 84, 93, 94], but using camera in outdoors has troubles such as illumination

issues. In our work we will concentrate in the use of 3D range sensing as the primary source of information for mapping, a technique better suited for unstructured outdoor mobile robot urban mapping [106], because it offers much more detailed information than 2D maps. The disadvantages of 3D sensing technologies are its slower acquisition rate and the significant increase in data that needs to be processed, calling for a compact and robust representation of the map.

For the purpose of this thesis we developed two 3D range sensors based on 2D range scanners. The first sensor is a tilting unit built with a RS4-Leuze 2D laser. The disadvantage of this sensor was that the acquisition rate was slow and that the resulting point cloud distribution was highly non-uniform. Furthermore, given the large weight of the Leuze Rotoscan, the tilting unit was mechanically unstable (i.e is like a pendulum), thus difficult to control. The second 3D range sensor we built included a Hokuyo UTM-30LX scanner mounted in a slip-ring, for which we filed a patent in 2012 [3]. The slip-ring allowed the sensor to be always rotating at a constant speed, thus avoiding the uneven distribution of sensed points from the acceleration and deceleration of the tilting unit. Moreover, by mounting the laser unit just above the rotating motor, it became very stable, thus easier to control compared with the tilting unit. Although these sensors were not as fast as commercial state-of-the-art range scanning systems when they were built, such as the Velodyne family, they not only provided a much denser point cloud than those sensors, but were also a much more affordable solution to our research group.

Multiple 3D scans are necessary to build a map with enough environment information for navigation. To create a correct and consistent map, the scans must have a common reference coordinate system. The process of aligning scans is called registration [14], and is usually done for a pair of scans at a time. This method is prone to error accumulation from local data alignment leading to an inconsistent global map, making the map useless for navigation, unless corrected. In this thesis we contribute with novel range registration mechanism for pairs of 3D range scans using point-to-point and point-to-line correspondences [128], and a hierarchical correspondence search strategy. For the minimization we adopt in this work a metric that takes into account not only the distance between corresponding points, but also the orientation of their relative reference frames [17]. We also propose FaMSA, a fast technique for multi-scan point

cloud alignment that takes advantage of the asserted point correspondences during sequential scan matching, where the point match history can be used to speed up the computation of new scan matches [129]. To properly propagate the model of the sensor noise and the scan matching, we employ a closed-form first order error propagation computation [33].

To build a globally consistent map we consider the probabilistic alignment of 3D point clouds using a delayed-state Extended Information Filter (EIF) [51]. In this thesis we adapt the Pose SLAM algorithm, a simultaneous localization and mapping (SLAM) algorithm, to the case of 3D range mapping [132, 140]. Pose SLAM is the variant of SLAM where only the robot trajectory is estimated and where sensor data is solely used to produce relative constraints between robot poses. Pose SLAM maintains a compact state representation by limiting the number of links and nodes added to the graph using information content measures and is agnostic to the sensor used. In this thesis we use the relative poses resulting from the registration of the point clouds as constraints that nourish Pose SLAM to create an overall map of the whole robot trajectory.

These dense mapping techniques are capable of producing impressively rich 3D environment models [140, 145]. Yet, these maps need still to be processed to identify traversable regions and to plan and execute navigation sequences. Region classification for off-road navigation is a challenging task. The main reasons being the irregularity of the surface and the need to account for the non-holonomic constraints of the vehicles.

State-of-the art terrain classification methods extract features from these dense point clouds [28], and use them to build traversable and non-traversable classes, usually through parametric classification. Feature thresholding is a difficult task that usually requires human intervention [54, 125, 135]. As an alternative, learning algorithms with manual data labelling [70] can also be used.

In this thesis we present a pair of methods to attain high-level off-line classification of traversable areas, in which training data is acquired automatically from navigation sequences[116]. Traversable features came from the robot footprint samples during manual robot motion, allowing us to capture terrain constrains not easy to model. Using only some of the traversed areas as positive training samples, our algorithms can successfully classify the rest of the traversable terrain. This is in contrast to other

techniques which search for ad-hoc features such as global planarity constraints which are not always in accordance with the training data.

Once the terrain has been classified as traversable or non-traversable, the last step is to plan and execute a trajectory from one location in the map to another. To plan safe trajectories we introduce in this thesis a path planner that guarantees reachability at a desired robot pose with significantly lower computation time than competing alternatives [130]. Our interest is to find safe trajectories that guarantee reachability satisfying the robot non-holonomic constrains. To search for the best path, our planner incrementally builds a tree using the A* algorithm [72, 76], and includes a hybrid cost policy to efficiently expand the search tree, combining random sampling from the continuous space of kinematically feasible motion commands with a cost to goal metric that also takes into account the vehicle non-holonomic constraints. The planer also allows for node rewiring, and to speed up node search, our method includes heuristics that penalize node expansion near obstacles, and that limit the number of explored nodes. The method book-keeps visited cells in the configuration space, and disallows node expansion at those configurations in the first full iteration of the algorithm. We validate the proposed method with experiments in extensive real scenarios from different very complex 3D outdoors environments, and compare it with other techniques such as the A*, RRT [73] and RRT* [56] algorithms.

The thesis is structured in two parts. The first part is devoted to the globally consistent map building, it includes the proposed scan matching method and the used variant of SLAM. The second part is devoted to the planning of robot navigation sequences for complex 3D environments, where we introduce the terrain classification method and the path planner. Figure 1.1 shows a block diagram representing the proposed architecture.

1.1 Summary of contributions

The contributions presented in this thesis constitute a step towards an integrated framework for mapping, terrain classification, traversability analysis, and path planning for autonomous mobile robots in 3D environments. These contributions can be listed as follows:

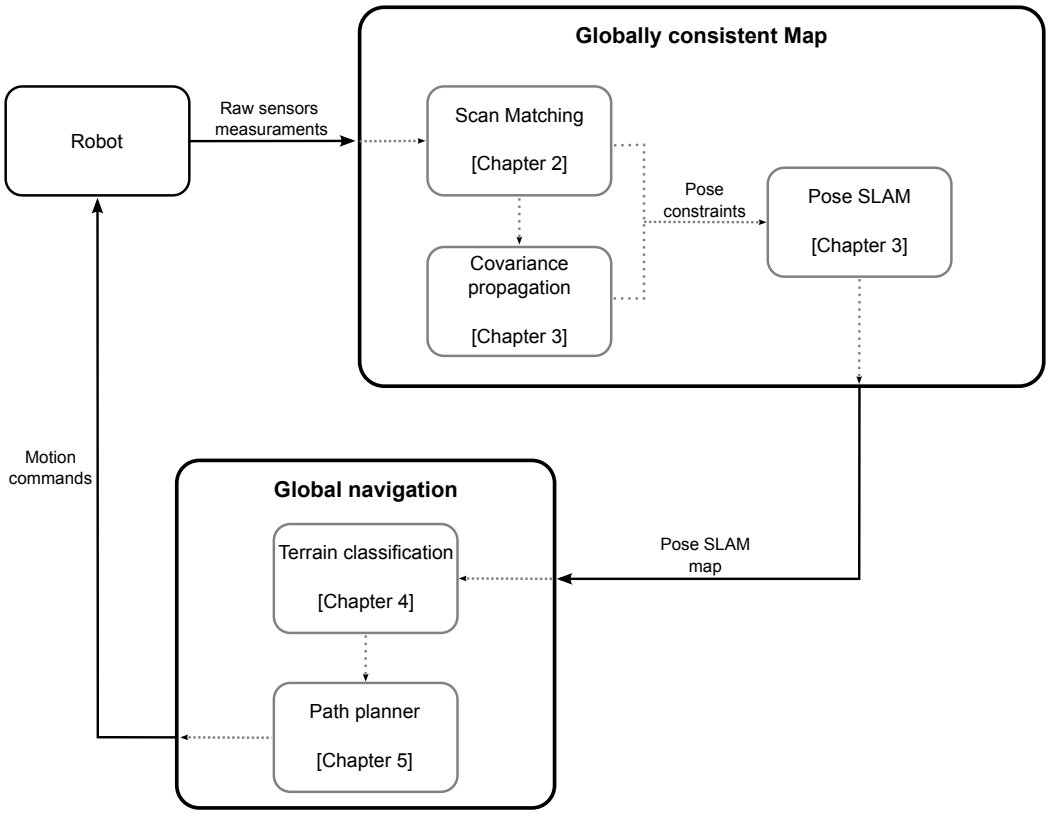


Figure 1.1: System architecture and thesis outline.

- *Accurate and fast range data registration during pairwise and multi-scan range data registration (Chapter 2).*

  We present a registration algorithm based on the well known iterative closest point algorithm. We choose to use a metric alternative to the Euclidean distance to perform the alignment [128]. We propose a new hierarchical correspondence search strategy, using a point-to-plane metric at the finest level and a point-to-point search at the coarsest level. To define the level we use a threshold to adjust error of the local fitted plane and its adjacent neighbors. Moreover, we introduce a novel technique for fast multi-scan point cloud alignment at loop closure that takes advantage of the asserted point correspondences during sequential scan matching [129].

- *Development and construction of range scanners to produce rich 3D point clouds (Chapter 2).*

  We built and patented a light weight and fast 3D range scan laser capable to work on-line in robot applications such as autonomous navigation [3]. This scanner superseded in computation time, and range resolution the previous scanner also developed in our group [87]. The sensor has a 360 degree horizontal view of the scene and a vertical field of view of 270 degrees. And, using these scanners as primary sensors, we adapted PoseSLAM, the information-based mapping solution previously developed in our group to the 3D mapping case [4, 132, 140]. The methodology was further used to build maps for the calibration of a camera network for the EU URUS project [100, 101], as well as for our own classification and path planning experiments.

- *Classification techniques for complex 3D environments (Chapter 4).*

  We present two techniques to detect and classify navigable terrain in complex 3D environments. This is a high-level off-line classification mechanism that learns traversable regions from larger 3D point clouds acquired with a laser range scanner. We approach the problem using Gaussian Processes as a regression tool, in which the terrain parameters are learned; and also for classification, using samples from traversed areas to build the traversable terrain class [116].

- *Path planning algorithm for 3D scenarios using point cloud (Chapter 5).*

  We propose a new randomized method to plan safe trajectories for complex 3D environments. The path planner guarantees reachability at a desired robot pose with significantly lower computation time than competing alternatives. The method is a modified A* algorithm that uses a hybrid node expansion technique that combines a random exploration of the action space meeting vehicle kinematic constraints with a cost to goal metric that considers only kinematically feasible paths to the goal. The method includes also a series of heuristics to accelerate the search time. These include a cost penalty near obstacles, and a filter to prevent revisiting configurations. We also allow node rewiring when revisiting previously explored robot configurations [130].

1.2 Publications derived from this thesis

[1] R. Valencia, **E.H. Teniente**, E. Trulls, and J. Andrade-Cetto. 3D mapping for urban service robots. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3076-3081, Saint Louis, October 2009. [140]

[2] A. Ortega, B. Dias, **E.H. Teniente**, A. Bernardino, J. Gaspar, and J. Andrade-Cetto. Calibrating an outdoor distributed camera network using laser range finder data. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 303-308, Saint Louis, October 2009. [100]

[3] J. Andrade-Cetto, A. Ortega, **E.H. Teniente**, E. Trulls, R. Valencia, and A. Sanfeliu. Combination of distributed camera network and laser-based 3D mapping for urban service robotics. In Proceedings of the IEEE/RSJ IROS Workshop on Network Robot Systems, pages 69-80, Saint Louis, October 2009. [4]

[4] **E.H. Teniente**, R. Valencia, and J. Andrade-Cetto. Dense outdoor 3D mapping and navigation with Pose SLAM. In Proceedings of the III Workshop de Robótica: Robótica Experimental, pages 567-572, Seville, 2011. [132]

[5] **E.H. Teniente** and J. Andrade-Cetto. FaMSA: Fast multi-scan alignment with partially known correspondences. In Proceedings of the European Conference on Mobile Robotics, pages 139-144, Orebro, September 2011. [129]

[6] J. Andrade-Cetto, M. Morta, P. Grosh, **E.H. Teniente**, Dispositivo medidor de distancia y magnitudes omnidireccional. Patent filed July 4th, 2012 (P201231044). [3]

[7] **E.H. Teniente** and J. Andrade-Cetto. HRA*: Hybrid randomized path planning for complex 3D environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1766-1771, Tokyo, November 2013. [130]

[8] A. Ortega, M. Silva, **E.H. Teniente**, R. Ferreira, A. Bernardino, J. Gaspar, J. Andrade-Cetto. Calibration of an Outdoor Distributed Camera Network with a 3D Point Cloud. Sensors 14.8 (2014): 13708-13729. [101]

[9] A. Santamaria, **E.H. Teniente**, M. Morta, and J. Andrade-Cetto. Terrain classification in complex 3D outdoor environments. Journal of Field Robotics 32.1 (2015): 42-60. [116]

Chapter 2

Point cloud registration

In this Chapter we discuss how to compute the rigid body transformation between two range images. If the range images are computed by the same sensor aboard a robot at two instances, then the transformation represents the relative motion between the two robot poses. To compute this transformation, we must align the two point clouds. The de-facto standard to compute this registration is the well-known iterative closest point algorithm (ICP) [14, 17, 20, 95, 117]. Starting with an initial guess for the transformation, the algorithm iteratively searches for point correspondences among the two point clouds and revises the transformation that minimizes the sum of distances between point matches.

In our implementation of ICP, we have chosen a metric that takes into account not only the distance between the points, but also the angle between their relative reference frames [17]. Moreover, given the large size of the point clouds treated and the amount of noise present in them, we have devised strategies point cloud sub-sampling and for outlier rejection during the correspondence search. We propose a new hierarchical correspondence search strategy, using a point-to-point search at the coarser levels and a point-to-plane metric at the finest level for improved accuracy. The level at which correspondence is taking place in this hierarchy is indicated by a user-selected threshold over the local plane fitting error. The proposed correspondence search strategy is validated on different data sets.

We also present in this chapter FaMSA, a technique for fast multi-scan point cloud alignment strategy that takes advantage of the asserted point correspondences during

sequential scan matching to aid in subsequent point match searches. The technique is especially relevant during loop closure for 3D SLAM. In FaMSA, the point match history is used to speed up the computation of new scan matches, hence allowing to quickly match a new scan with multiple consecutive scans at a time, with the consequent benefits in computational speed. The registration error is shown to be comparable to that of independent scan alignment. Results are shown for dense 3D outdoor scan matching.

The scan matching methods presented in this chapter represent the basis for the computation of relative motion constraints used as inputs for the SLAM method described in the next chapter.

2.1 Related work

Scan matching algorithms are often used in mobile robotics to correct the relative motion of a vehicle between two consecutive configurations, by maximizing the overlap between the range measurements obtained at each configuration. The most popular scan matching methods in [114] are based on the ICP algorithm [14] which is borrowed from the computer vision community. The objective of this algorithm is to compute the relative motion between two data sets partially overlapped. The algorithm iteratively minimizes the Mean Square Error (MSE) of point correspondence distances and proceeds as follows: first, for each point in one data set, the closest point in the second one is found or vise versa (correspondence step), then the motion that minimizes the MSE between the correspondences is computed (registration step), finally the data shape is updated (update step).

In the registration proposed by Besl and Mckey a point-to-point metric is used to measure the "closeness" of data, they also suggest an accelerated version of the algorithm by using a linear approximation and a parabolic interpolation with the last three minimization vectors if they are well aligned, which means they have been moving in an approximately constant direction. The use of sampling or tree-based-search to speed up the algorithm are mentioned as future refinements to reduce the computational cost. In [20] the authors propose a point-to-plane error metric, which makes the algorithm less susceptible to local minima than the metric proposed in [14]. The idea

in [20] is that given a so called control point in the first surface, the distance to the nearest tangent plane in the second cloud is computed. In [18] the authors suggest a point-to-projection solution computing the sum of the distances between the two range views in the direction of the rays. This has also been called "reverse calibration". This approach makes registration very fast because it does not involve any search step to find the correspondences. However, one of its disadvantages is that the resulting registration is not as accurate as the one given in the point-to-point and point-to-plane metrics [114]. In [138] the authors propose a point-to-triangle correspondence using meshes, which they call zippered meshes, finding the nearest position on a mesh to each vertex on any other mesh. They find the rigid transformation that minimizes a weighted least-squared distance between correspondences with a value in the range from 0 to 1 called confidence. For the case of structured light scanners, they measure the confidence of a point on a mesh to be the angle between the mesh normal and the sensor viewing angle. The confidence is a measure of how certain they are of a given range point's position, which helps to eliminate possible wrong matches. In [22] the authors present a robustified extension of the ICP applicable to overlaps under 50%, robust to erroneous and incomplete measurements, and with easy-to-set parameters called Trimmed ICP (TrICP). The algorithm is based on the consistent use of the Least Trimmed Squares algorithm [113] in all phases of the operation. On the other hand, in [147], genetic algorithms are used to maximize an objective function, where the genes are formed by concatenating six binary coded parameters, representing the three angles of rotation and the 3-DOF for translations. A new error metric for the 2D space which explores the compensation between the rotation and translation was proposed in [82, 83]. Later in [16, 17] this metric is extended to the 3D space. The authors used a point-to-projection minimization strategy using triangles as the projection surface, and performing Nearest Neighbor (NN) correspondence search in the space of the new metric. They did not tackle the computational complexity of the algorithm in any way, so it is understood that they used brute force search to do the matching.

The computational bottleneck of ICP concentrates during the search for point matches. One strategy to reduce such computational complexity is to use tree-based search techniques [102]. In [95] for instance, the authors used a library called Approximate Nearest Neigborh (ANN), developed by [8], that includes either a balanced kd-tree or a box decomposition tree (bd-tree). Nütcher et al. [95] showed that kd-trees

are faster than bd-trees for the NN problem. In [122] the authors used a kd-tree with a caching technique, where in the first iteration, n-neighbors for each point in the reference cloud are extracted from the model query and cached. It is assumed that after updating the reference cloud, the majority of cached points for each point in the updated cloud should remain as neighbors and hence need not be matched again. In [12] a double z-buffer structure is used which provides an explicit space partitioning. While in [147] it is said that the time matching can be significantly reduced by applying a grid closest point (GCP) technique, the GCP turns out to be another sampling scheme which consists of superimposing a 3D fine grid on the 3D space such that the two clouds lie inside the grid, dividing the 3D space into cells, each cell with an index of its closest point in the model set. In [41] the authors present a new solution to the NN search problem for the matching step, which they called Spherical Triangle Constraint. The Spherical Constraint is applied to determine whether or not the nearest neighbor falls within the neighborhood of each point estimate, and if so, the Triangle Constraint and the Ordering Theorem are applied to the neighborhood to quickly identify the correspondences. The Ordering Theorem orders a set of points by increasing distance to some point. The authors showed that after approximately 20 iterations, their method becomes more efficient than kd-trees in computational complexity and time execution. Akca and Gruen [1] used a box structure [21] which partitions the search space into boxes. For a given surface element, the correspondence is searched only in the box containing this element and in the adjacent boxes. The correspondence is searched in the boxing structure during the first few iterations, and in the meantime its evolution is tracked across iterations. In the end, the search process is carried out only in an adaptive local neighborhood according to the previous position and change of correspondence. One of the main advantages of the box structure is that it has a faster and easier access mechanism than the one provided by tree-based search methods.

Another common strategy to accelerate the matching process is to reduce the number of points. Sampling the data reduces the match execution time by a constant factor, but retains the same asymptotic computational complexity. The techniques used for sampling data vary. Coarse-to-fine strategies have been used by [149] and [138]. They start using a less detailed description of the data and as the algorithm approaches the solution, the resolution is hierarchically increased. In [138] and [18] on the contrary, the authors used uniform sampling. And in [81], the authors used random sampling

at each iteration. Moreover, [98] and [43] used a technique best suited to the nature of data for laser range finders, the so called reduction filter, which has been shown to work well for realtime applications.

However, sampling methods are very sensitive to data content, i.e. noise level, occlusion areas, complexity of the range data, etc. If too many points come from outliers due to sensor errors, this may produce too many wrong correspondences, and may cause the solution to converge on a local minimum leading to a poor final overlap, or in the worst case, to divergence. We shall remember that the original algorithm from Besl and Mackay considers data sets without outliers. Several approaches to dismiss possibles outliers have been proposed using rejection strategies. Rejections based on thresholds for the maximum tolerable distance between paired points were implemented by [138], the threshold is set as twice the maximum tolerable space between range point meshes and is adaptively changed when building the mesh. A statistical rejection method based on the distribution of point distances was used by [149] and [109]. They used two thresholds for the maximum allowed distance between paired points, one of them dynamic. In [81] the authors reject pair matches whose point-to-point distance are larger than some multiple of their standard deviation. Pulli, as well as [138], used not only statistical reasoning, but also topological information to discard matches. They removed matches that occur on mesh boundaries, and in [149] and [109] they used the angle between the normals of the paired points as a constraint to keep matches. Specifically for laser range data, the authors in [98] employed a median filter to remove the Gaussian noise for each scan row.

The uniformity of the point cloud plays also a very important role during registration. and it is linked to the process used to acquire the point cloud. Initial attempts to generate 3D point clouds for mobile robot navigation considered the use of 2D laser scanners rigidly attached to the mobile robot, and the aggregation of a point cloud by sweeping across the scenario whilst moving the robot [47]. This methodology made the point cloud quality dependent on the quality of the platform odometry. An alternative was to mount the 2D laser atop a servo mechanism, in either a pan or a tilt configuration for indoors [127, 144] or for outdoors [96, 98, 140]. In this case, the robot remains still while a 3D snapshot is being acquired by the scanning mechanism, with the consequence of significantly reduced error in the point cloud its, but at the expense

of intermittent robot motion. For continuous scanning, Lamon [71] presented a rotational system using two lasers that produces a full 360° vertically scanned scenario. This configuration is similar to our own scanning configuration [3, 132].

2.2 Range image registration

The main idea of the range image registration, also known as scan matching, is to compute the relative motion between two consecutive snapshots of a scene with a dense 3D sensor. The technique is local in nature, but is typically used in mobile robotics to update global localization and mapping estimates. In this section, we described in detail the range image registration principles.

Given a point set $P = \{\mathsf{p}_1, \mathsf{p}_2, ..., \mathsf{p}_n\}$ acquired from a reference frame and a second point set $P' = \{\mathsf{p}'_1, \mathsf{p}'_2, ..., \mathsf{p}'_m\}$, acquired from a new frame; we need to estimate the relative displacement $\mathrm{x} = (x, y, z, \psi, \theta, \phi)$ between the reference frame and the new frame. The Iterative Closest Point (ICP) algorithm deals with this problem in an iterative process [14]. Starting from an initial estimate of the displacement x_0, at each iteration it establishes a set of correspondences between the two point sets and updates the relative displacement x_k between them by minimizing the error of the correspondences. The process is repeated until convergence.

The basic idea in the correspondence step is to find the closest point in one point set for each point in the other one. Closeness, lower than a given threshold d_{min}, is measured according to a specific metric, usually the Euclidean distance. This search is also known, and from here referenced, as the Nearest Neighbor (NN) search. If we refer with $\mathrm{x}_k(\mathsf{p}'_j)$ to the rototranslation of point p'_j in P' by x_k to the reference frame of P then, the above described point correspondence is

$$c_i = \arg\min_{\mathsf{p}'_j} \{ d(\mathsf{p}_i, \mathrm{x}_k(\mathsf{p}'_j)) < d_{min} \} \ . \tag{2.1}$$

The result is, for each iteration k, a set of l correspondences $C_k = \{(\mathsf{p}_i, c_i) | i = 1 \dots l\}$.

Besl and McKey [14] assume that for each point in the reference set there must exist a correspondence in the data set, in our application this is not the case, hence $l \leq n$ and $l \leq m$.

The metric to establish the correspondences computes the distance between the two points. It is normally defined by the L2 norm (Euclidean distance)

$$d(\mathsf{p}, \mathsf{p}') = \left(\sum_{1 \leq d \leq D} (p_d - p'_d)^2 \right)^{1/2} \tag{2.2}$$

where $D = 3$ is the dimension size of the data we are working with, i.e., $\mathsf{p} = [p_x, p_y, p_z]$.

An alternative to the Euclidean distance, is the metric proposed in [16, 17], where the relative displacement, is expressed as a 3D rigid body transformation defined by a vector $\hat{\mathrm{x}} = (x, y, z, \theta n_x, \theta n_y, \theta n_z)$, it represents the position and orientation ($-\pi < \theta < \pi$) of the range finder laser. The norm of $\hat{\mathrm{x}}$ is defined by

$$\|\hat{\mathrm{x}}\| \;=\; \sqrt{x^2 + y^2 + z^2 + l_m^2 \theta^2} \tag{2.3}$$

with $l_m \in \mathbb{R}^+$ for a distance.

Given two points $\mathsf{p} \in P$ and $\mathsf{p}' \in P'$, using this metric, the distance between both points is

$$d(\mathsf{p}, \mathsf{p}') \;=\; \min \left\{ \|\hat{\mathrm{x}}\| \;\middle|\; \hat{\mathrm{x}}(\mathsf{p}') = \mathsf{p} \right\} \tag{2.4}$$

where

$$\hat{\mathrm{x}}(\mathsf{p}') = R(\mathbf{n}, \theta)\mathsf{p}' + \mathbf{t} \tag{2.5}$$

with $\mathbf{t}$ the translation vector (x, y, z), $R(\mathbf{n}, \theta) = \mathbf{I} + U(\mathbf{n}) \sin(\theta) + (U(\mathbf{n}))^2(1 - \cos(\theta))$ the Rodrigues representation of the matrix rotating an angle θ about the unit vector $\mathbf{n}$, and $U(\mathbf{n}) = \mathbf{n}_\times$ the so-called cross-product matrix of $\mathbf{n}$. Unfortunately there is no closed form expression for d with respect to the coordinates of the points. However a valid approximation could be computed with small rotations. Linearizing Eq. (2.5) about $\theta = 0$ leads to $\cos(\theta) \approx 1$ and $\sin(\theta) \approx \theta$, hence $R(\mathbf{n}, \theta) \approx I + U(\mathbf{n})\theta$, and we can express Eq. (2.5) as

$$\hat{\mathrm{x}}(\mathsf{p}') \approx \mathsf{p}' + U(\mathbf{n})\theta\mathsf{p}' + \mathbf{t} \tag{2.6}$$

and from the skew symmetric matrix properties, $U(\mathbf{n})\theta\mathsf{p}' = -U(\mathsf{p}')\mathbf{n}\theta$, the above equation could be rewritten as

$$\hat{\mathrm{x}}(\mathsf{p}') \approx \mathsf{p}' + U(\mathsf{p}')r + \mathbf{t} \tag{2.7}$$

with $r = \mathbf{n}\theta$.

If Eq. (2.4) squared is further developed with this linear approximation, then we have that:

$$d^{ap}(\mathsf{p},\mathsf{p}')^2 = \boldsymbol{\delta}^T M \boldsymbol{\delta} \tag{2.8}$$

$$= \|\boldsymbol{\delta}\|^2 - \frac{\|\mathsf{p}' \times \boldsymbol{\delta}\|^2}{k} . \tag{2.9}$$

and, working out this squared value, the approximate distance is obtained

$$d^{ap}(\mathsf{p},\mathsf{p}') = \sqrt{\|\boldsymbol{\delta}\|^2 - \frac{\|\mathsf{p}' \times \boldsymbol{\delta}\|^2}{k}} \tag{2.10}$$

where $k = \|\mathsf{p}'\|^2 + l_m^2$, $\boldsymbol{\delta} = \mathsf{p} - \mathsf{p}'$ and l_m is the trade-off between translation and rotation, and is user specified. Be aware that, if $l_m \to \infty$ the new distance becomes the Euclidean distance.

In the registration step, the relative displacement between the two point sets is solved using a last squares minimization over this approximate distance [17], computing the rigid body transformation that minimizes the displacement. The accumulated distance for all points in the cloud becomes

$$E_{dist}(\hat{\mathrm{x}}) = \sum_{i=1}^{l} d^{ap}(\mathsf{p}_i, \hat{\mathrm{x}}(\mathsf{p}'_j))^2 \tag{2.11}$$

with j the index of the point cloud in P' closest to the p_i in P. From Eq. 2.8, the above equation could be rewritten as

$$E_{dist}(\hat{\mathrm{x}}) = \sum_{i} \boldsymbol{\delta}_i^T(\hat{\mathrm{x}}) M_i \boldsymbol{\delta}_i(\hat{\mathrm{x}}) \tag{2.12}$$

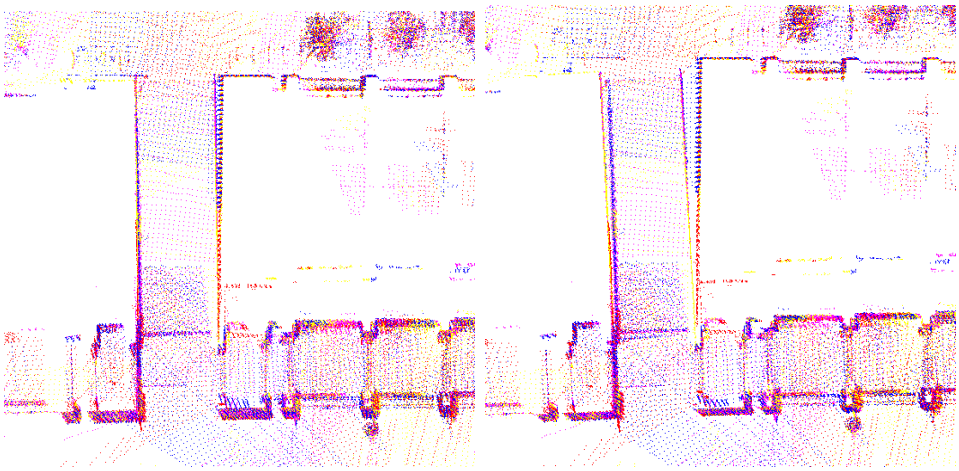
and

$$\begin{aligned} \boldsymbol{\delta}_i(\hat{\mathrm{x}}) &= \mathsf{p}_i - \hat{\mathrm{x}}(\mathsf{p}'_j) \\ &\approx \mathsf{p}_i - \mathsf{p}'_j + U_j r - \mathbf{t} \end{aligned} \tag{2.13}$$

with :

$$M_i = \begin{pmatrix} \mathsf{p}_{ix}^2 + l_m^2 & \mathsf{p}_{ix}\mathsf{p}_{iy} & \mathsf{p}_{ix}\mathsf{p}_{iz} \\ \mathsf{p}_{ix}\mathsf{p}_{iy} & \mathsf{p}_{iy}^2 + l_m^2 & \mathsf{p}_{iy}\mathsf{p}_{iz} \\ \mathsf{p}_{ix}\mathsf{p}_{iz} & \mathsf{p}_{iy}\mathsf{p}_{iz} & \mathsf{p}_{iz}^2 + l_m^2 \end{pmatrix} \tag{2.14}$$

$$U_j = U(\mathsf{p}')_j = \begin{pmatrix} 0 & -\mathsf{p}'_{jz} & \mathsf{p}'_{jy} \\ \mathsf{p}'_{jz} & 0 & -\mathsf{p}'_{jx} \\ -\mathsf{p}'_{jy} & \mathsf{p}'_{jx} & 0 \end{pmatrix} \tag{2.15}$$



(a) (b)

Figure 2.1: Comparison of ICP metrics. a) Metric based, $l_m = 30$. b) Euclidean distance.

The rigid body transformation that minimizes equation (2.12) is given by

$$\hat{\mathrm{x}}_{\min} = \left(\sum_{i=1}^{l} \begin{pmatrix} M_i & -M_i U_j \\ -U_j^T M_i & U_j^T M_i U_j \end{pmatrix} \right)^{-1} \cdot \sum_{i=1}^{n} \begin{pmatrix} M_i \boldsymbol{\delta}_i \\ U_j M_i \boldsymbol{\delta}_i \end{pmatrix} . \tag{2.16}$$

Figure 2.1 shows a sample application of this rotation aware metric. The frame to the left shows the results of the alignment of two point clouds with a weight factor of $l_m = 30$. The frame to the right shows alignment results of the same point clouds using the Euclidean distance as metric.

2.3 A hybrid hierarchic ICP

We propose a variant of the ICP algorithm that uses local planar information to decide the correspondence search strategy to use. To find the correspondences several metrics are available [114]: point-to-point [14, 98, 122], point-to-plane [20, 149], or point-to-projection with triangular surfaces [16–18]. In our method we use a combination of these in hierarchy for the correspondence search step, using a point-to-point metric at the coarsest levels and a point-to-plane strategy at the finest level. For the minimization step, we adopt the metric proposed in [17], but we use point-to-point matching

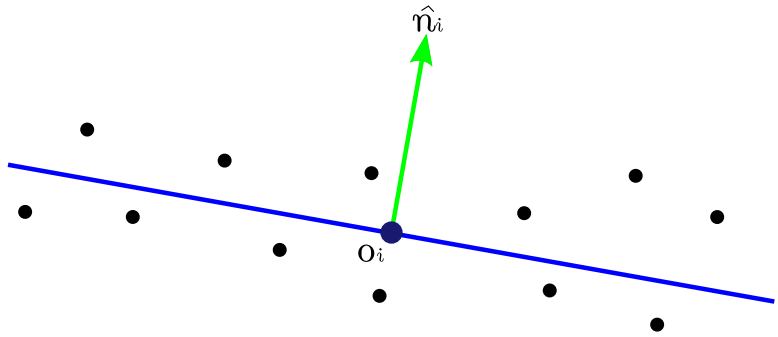


Figure 2.2: Plane fitting and representation by the mean point o_i and its normal $\hat{\mathbf{n}}_i$.

instead a point-to-mesh matching, thus avoid the computational burden of estimating the corresponding triangle in the mesh.

2.3.1 Metrics for data association

We have already stated that point-to-point correspondence is the basic data association metric used in ICP, and explained in the previous section how to compute it. We now concentrate on explaining the point-to-plane correspondence search [20, 149]. Given its higher computational cost than the point-to-point metric, we only use it at the finest level of our hierarchical ICP scheme. The point to plane data association proceeds as follow: given a 3D point cloud P containing k individuals points $(\mathsf{p}_1, \mathsf{p}_2, ..., \mathsf{p}_k)$, we compute local planar patches T_p for each individual point. Next, for each point p' in a query point cloud P' we find the plane T_p for which the projection of p' onto it is the closest. This projection is the corresponding point.

The local planes are computed fitting the set of nearest neighbors in P to each point p in a given search radius. This neighbor set is called P_{T_p}. We represent the local planar path T_p with a point o on it, and a unit normal vector $\hat{\mathbf{n}}$ that indicates the plane's orientation [5], see Fig. 2.2. Local planar patches are only computed if a point has at least k-NN in the given search radius. Otherwise the point is removed from the point cloud since it does not offer sufficiently relevant information and can be considered spurious.

The point o is computed finding the mean of P_{T_p}. The unit normal vector $\hat{\mathbf{n}}$ is the eigenvector associated to the smallest eigenvalue of the correlation matrix M defined

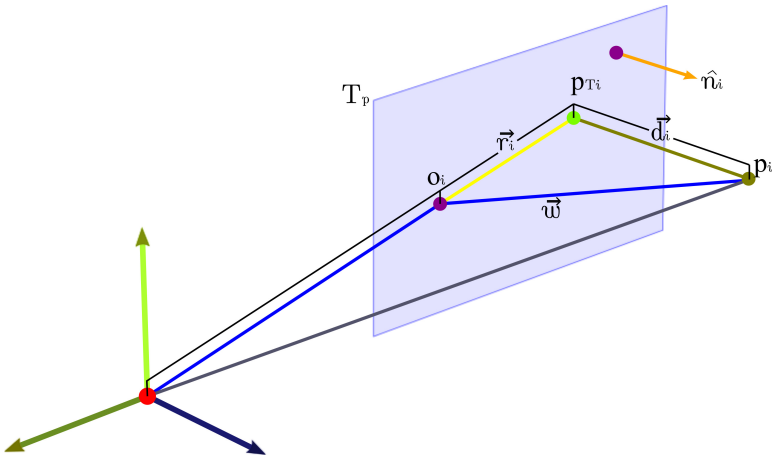


Figure 2.3: Query point p_i and its closest point $\mathsf{p}_{T,i}$ in the plane.

by

$$M = Q - \frac{\mathsf{p}_q \mathsf{p}_q^T}{k^2}$$

where :

$$\mathsf{p}_q = \sum_{i=1}^{k} \mathsf{p}_i \qquad \forall \qquad \mathsf{p}_i \in P_{T_p}$$

$$Q = \sum_{i=1}^{k} \mathsf{p}_i \mathsf{p}_i^T$$

then

$$\hat{\mathbf{n}} = eig(M)$$

We want to find p_{T_p}, the closes point to p that lies on the plane T_p. To find this projected point, it is necessary to find the minimum distance d between the computed tangent plane T_p and p, where the signed distance is defined as $d = (\mathsf{p} - o)^T \hat{\mathbf{n}}$. Let us define the distance vector as $\mathbf{d} = d^T \hat{\mathbf{n}}$, and from Fig. 2.3:

$$\mathbf{w} = \mathsf{p} - o$$

$$\mathbf{r} = \mathbf{w} - \mathbf{d} \tag{2.17}$$

$$\mathsf{p}_{T_p} = o + \mathbf{r} \tag{2.18}$$

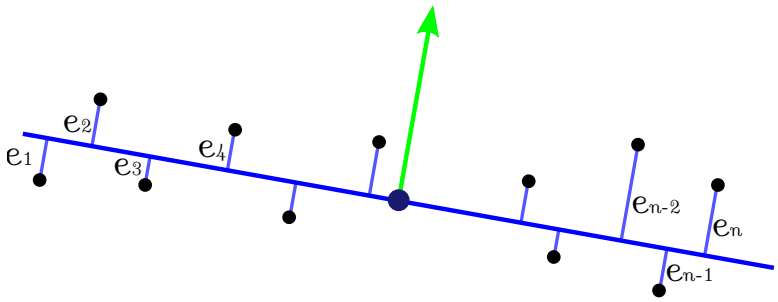


Figure 2.4: Plane with the errors of its adjacent points.

Substituting Eq. (2.17) on (2.18) and solving

$$\begin{aligned} \mathsf{p}_{T_p} =& o + \mathbf{w} - \mathbf{d} \\ =& \mathsf{p} - \mathbf{d} \end{aligned} \tag{2.19}$$

Where p_{T_p} is the correspondent nearest neighbor to p.

2.3.2 Hybrid hierarchic ICP algorithm

We now explain the rationale used to decide whether to use the point-to-point or point-to-plane metric. The idea behind is that, a point-to plane metric is expected to be more accurate than the point-to-point metric. But it is not only more expensive to compute, but also the support plane has to be sufficiently good. Then, we choose as level or threshold that indicates when to use one metric or the other the mean square error of the fitted local plane computed in Sec. 2.3.1. If the fitting plane error is larger than a given threshold, we consider that the plane does not have sufficient support or that it is too noisy for the point-to-plane metric to be used, and the point-to-point metric is used instead.

We compute this error as the average of each point in P_{T_p} to the plane T_p. See Figure 2.4,

$$e_{T_p} = \frac{1}{k} \sum_{i=1}^{k} \sqrt{((\mathsf{p}_i - o)^T \hat{\mathbf{n}})^2} \tag{2.20}$$

The following pseudo-code (Alg. 1) resumes our Hybrid Hierarchic Iterative Closest Point algorithm.

ICP($\mathrm{x}_{i+1}, P, P', \mathrm{f}$)
INPUT:
x_{i+1}: Relative robot displacement at time $i+1$.
P: Point cloud at time i.
P': Point cloud at time $i+1$.
f: Point clouds features.
OUTPUT:
x_{i+1}: Corrected relative displacement at time $i+1$.

1: **for** $k = 1$ to K **do**
2: $Y \leftarrow$ GETCORRESPONDENCES(P, P', f)
3: $e_k^2 \leftarrow$ COMPUTEERROR(Y)
4: $\hat{\mathrm{x}}_{min} \leftarrow$ MINIMIZATION(Y)
5: $(P', Y', \mathrm{x}_{i+1}) \leftarrow$ UPDATE($\hat{\mathrm{x}}_{min}, \mathrm{x}_{i+1}, P', Y$)
6: $d_k^2 \leftarrow$ COMPUTENEWERROR(Y')
7: **if** CONVERGENCE(e_k^2, d_k^2) **then**
8: RETURN ($\hat{\mathrm{x}}_{i+1}$)
9: **end if**
10: **end for**

Algorithm 1: Hybrid Hierarchic ICP.

In our proposed method, one of these three criterion must be reached for convergence: a desired maximum number of iterations, desired difference between the MSE error at the beginning of the iteration, and the MSE at the end, *i.e* $d_k^2 - e_k^2$ or a minimum d_k^2.

Note that we could have decided to use only the point-to-plane metric, and clean up tangent planes from outliers during fitting. This however is not only computationally more expensive, but also would result in missing relevant point-to-point feature correspondences at the coarser levels on non-planar regions, such as trees or bushes. Another reason to use our hierarchical scheme is to alleviate the possibility of falling into local minima, a problem common to most other ICP approaches.

Some of the correspondences can eventually come from wrong matches because of noise data or zones with low information. For this reason during the NN search we apply several filters. In the original ICP algorithm of Besl and McKey [14] it is assumed that for each point in the reference set there must be a correspondence in the query set. When using point clouds coming from real robots with real sensors, this is surely not the case, and adequate similarity tests must be implemented. Using point distance as the only criteria for point similarity usually leads to wrong data association and local minima. We do not want to compute distinctive feature signatures for each point, since these are computationally expensive to compute. Hence, we use as in [109], the oriented

normal similarity constraint, together with a statistical constraint [80], i.e, points at distances larger than a multiple of their standard deviation are rejected. Correspondence uniqueness is also enforced and to implement it we programmed appropriate bookkeeping of matches at every iteration.

2.3.3 3D range data filtering and data reduction

Point clouds are usually noisy and plagued with various artifacts of different kinds. Blurring could occur for example at the edges, where the laser beam of a particular scan hits two surfaces resulting in an erroneous data value. Reflections on mirrors and refractions of the laser beam through glass also produce annoying artifacts. We need to come up with efficient mechanisms to deal with all these situations since a few outliers not properly handled may lead to multiple wrong correspondences during matching, leading to an incorrect alignment.

On such heuristic used in [98] is to smooth the data. To that end, a median filter can be used [43], capable of recognizing noise in a scan and to replace it with a suitable measurement. Median filtering in our case proceeds as follows. A window of k points is gathered from the last point cloud around each scan point p_i. The points inside the window are sorted according to their range value, and p_i is replaced with the median value in the window, see Fig. 2.5(b). Since large window widths would most likely end up distorting the scan, typical values for median filter windows are rather small, a value of $k = 5$ has proved to work well in our case.

A second strategy is to reduce the number of points, this time to improve execution time for very dense point clouds as in our case. Sampling strategies must preserve the geometric nature of the surfaces [114], and in our case we use uniform sampling. To sample the point cloud uniformly, the space is split in voxels. Initially the corners of the volume enclosing the point cloud are found searching for the maximum an minimum value for each coordinate axis plus a small increment δ, i.e., $\mathbf{p}_{min} = (x_{min} - \delta, y_{min} - \delta, z_{min} - \delta)^T$, and $\mathbf{p}_{max} = (x_{max} + \delta, y_{max} + \delta, z_{max} + \delta)^T$. Then we compute for a desired voxel size vox_{size}, the total number of voxels needed for each axis $\lceil(\mathbf{p}_{max} - \mathbf{p}_{min})/vox_{size}\rceil$. Next, the right index of the corresponding voxel for each point in the cloud is computed with $\lceil(\mathbf{p}_i - \mathbf{p}_{min})/vox_{size}\rceil$, and the median point of the voxel is recorded.

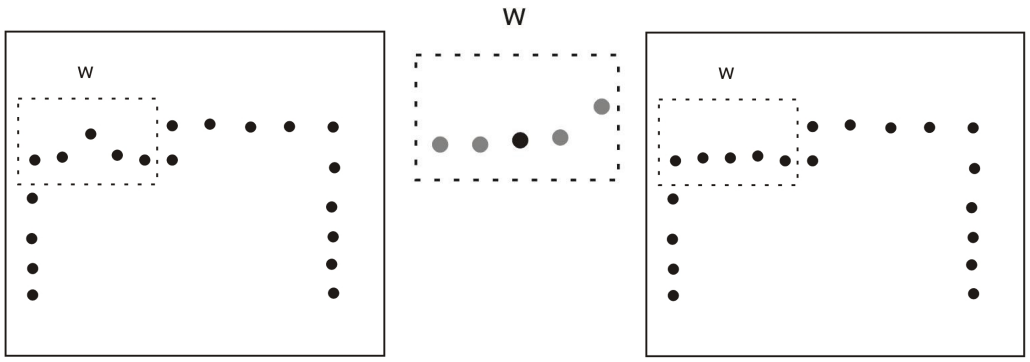
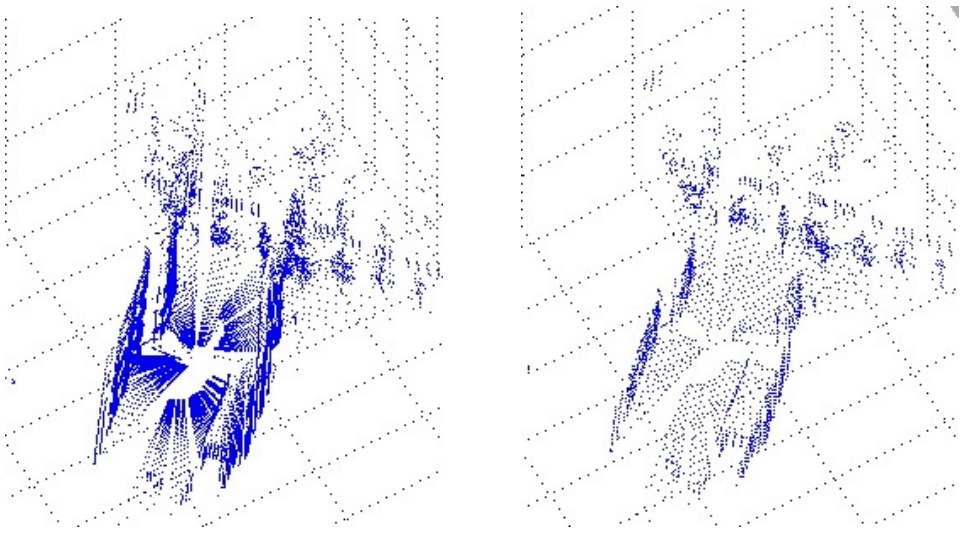


Figure 2.5: Median filter process. a) Selecting the window. b) Median point in black. c) Result of the median filter.

Finally, another technique to reduce the number of points is to limit the maximum distance for each local coordinate, this strategy also contributes to remove outlier readings because points far from the center are sparse and induce larger corrections to the ICP registration. Another reason to apply these limits is to weight the contribution between points along the vertical and horizontal planes. We have observed that if one of both surfaces has a significantly greater amount of points, that could lead to misalignment. The result of applying the mentioned reduction strategies is shown in Fig. 2.6(b).

2.3.4 Hybrid hierarchic ICP comparison

In the previous section we have stated the tools for our ICP proposal. To validate the method we show results on two different datasets acquired in the Campus Nord of the Universitat Politècnica de Catalunya, at the Barcelona Robot Lab (BRL), Fig. 2.7. This environment is equipped with a distributed camera network and full wireless coverage, available for your experiments. BRL was setup as part of the URUS project [115], covering an area over 10,000 square meters with several levels and underpasses, poor/intermittent GPS coverage, moderate vegetation, several points with aliasing, large amounts of regularity from building structures, sunlight exposure severely subject to shadows, and some dynamic objects e.g. walking people. It is the perfect scenario to subject our mapping algorithms to real world conditions.



(a) (b)

Figure 2.6: Sampling and pruning example. a) Original point cloud. b) Pruned and sampled point cloud.

We gathered our first dataset using our robot "Helena" (Fig. 2.8(a)), an Activmedia Pioneer 2AT robotic platform with a 3D range laser mounted atop, GPS and compass. The laser is a proprietary system designed at IRI [87], similar to the ones in [77, 98, 144]. It includes a 2D range laser sensor for pan readings mounted on a motorized structure giving the tilt movement. The 2D range sensor is a RS4-Leuze 2D laser, with a maximum resolution of 528 points per scan line, in a 190.08° ($-5.04° < \theta_{laser} < 185.04°$) amplitude. This means that every 0.36° a point is obtained. Each scan is about 76,000 points. In this case the odometry coming from the Activmedia platform was used to get a first estimate of the map.

Our second dataset was gathered by the ETHZ's Smartter robot [71], also at BRL during the experiment sessions of the project URUS. The range scanner system in the Samartter robot consists of two 2D Sick LMS291-S05 lasers rotating around a vertical axis (pan rotation) Fig. 2.9(b), delivering point clouds with a 360° vertical coverage. Each second, a full 3D scan of the environment around the vehicle was acquired, each scan with between 5,000 and 20,000 points, depending on the desired resolution, see Fig. 2.9(a). In this case, the first estimate of the vehicle trajectory fused wheel odometry, a differential GPS, an optical gyro and an inertial measurement unit.

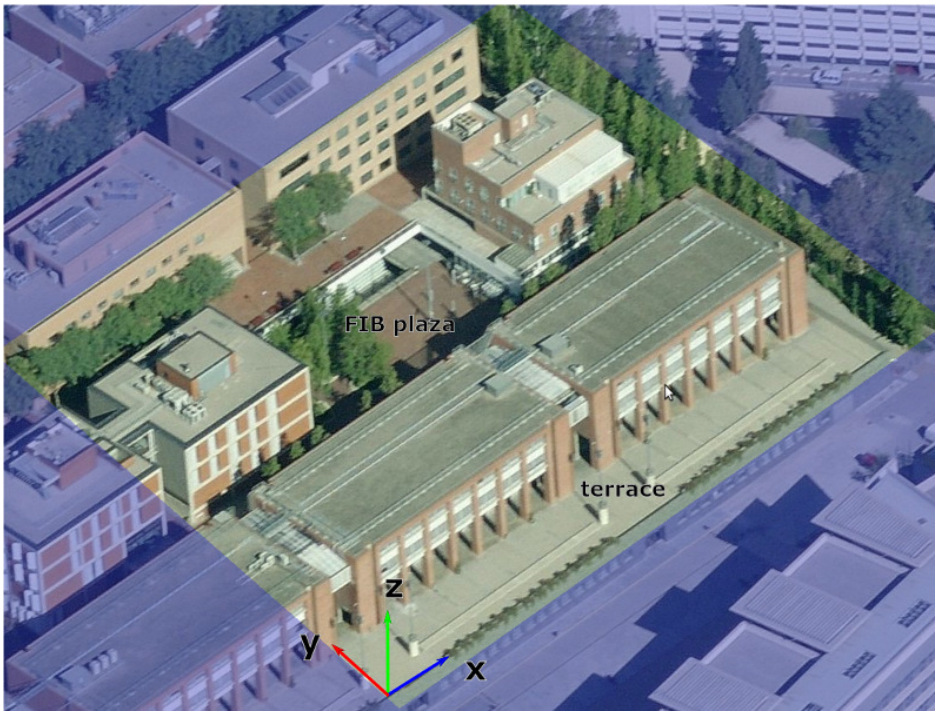


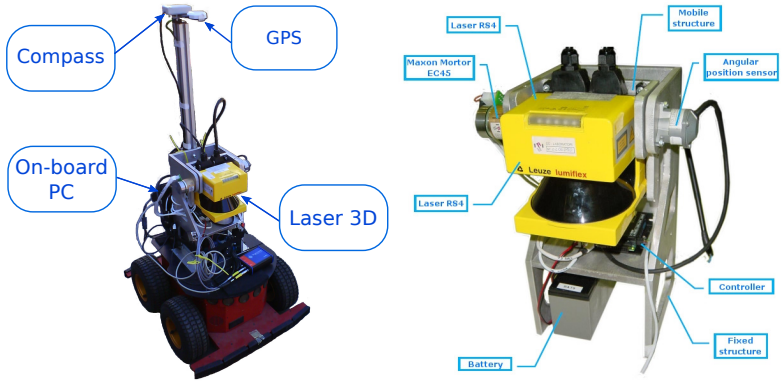
Figure 2.7: The Barcelona Robot Lab at the Campus Nord of the Universitat Politècnica de Catalunya.

We sampled and filtered both datasets as stated in Sec. 2.3.3. We show the values for each dataset in the Table 2.1. All parameters were set up experimentally except for the median filter value which was set to 5 following the suggestion in [43].

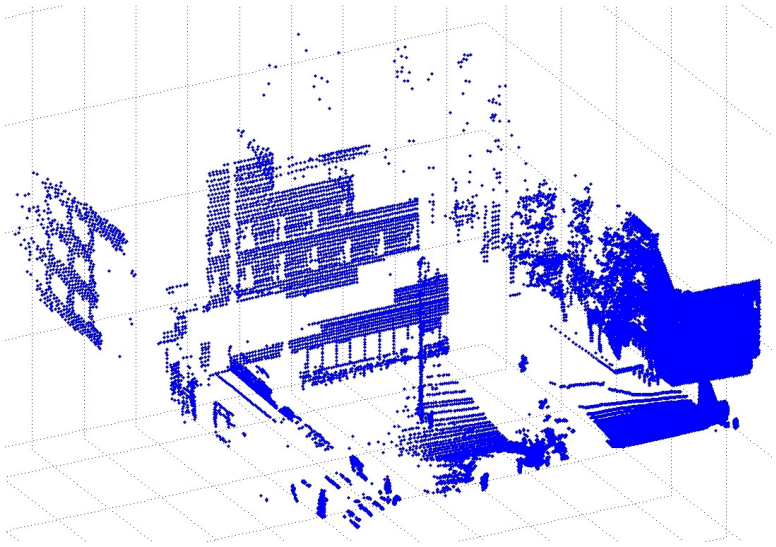
Dataset	vox_{size}	x_{max}, y_{max} (m)	z_{max} (m)
Smartter	0.45	$\pm$ 23	8
Helena	0.35	$\pm$ 25	9

Table 2.1: Sampling values for the data.

For both datasets we performed an empirical comparison of the different correspondence search strategies: point-to-point, point-to-plane and the hybrid correspondences search using two different l_m values, $l_m = \infty$ and $l_m = 50$. To compute the local fitted planes we use the 12-NN from the original point cloud to the query point. Using the unsampled point cloud to compute the local planes helps to preserve the local features for the registration. For Helena's dataset we decided experimentally to set



(a) (b)



(c)

Figure 2.8: a) IRI's Helena robot and its sensors. b) Our first custom built 3D range sensor. c) Example of an acquired scan.

(a) (b) (c)

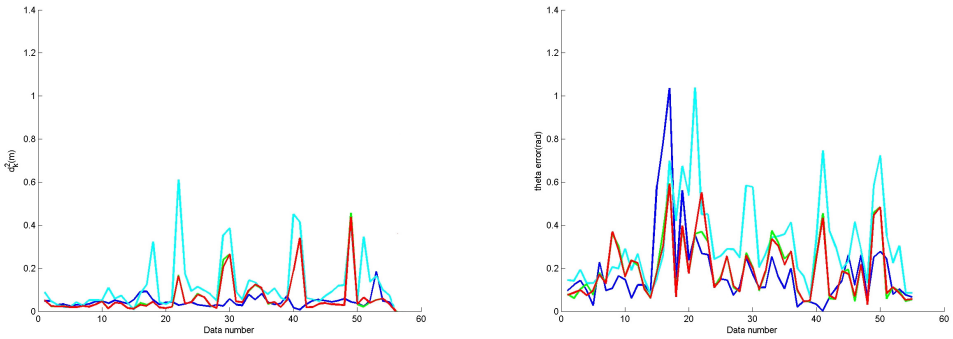
Figure 2.9: a) ETHZ's Smartter robot and its sensors. b) The mounted 3D scanner. c) Example of an acquired scan.

the following parameters, a maximum of 20 iterations, 92% accepted data, a pairing distance threshold of $3.3m$, 0.05 error threshold between e_k^2 and d_k^2, a minimum error of $d_k^2 = 0.065m$ for the convergence parameters, and in the hybrid case, a maximum tangent plane error of 0.12 m. For Smartter's dataset we also decided experimentally the algorithm parameters: a maximum of 25 iterations, a filtering of 10% of the data, a pairing distance filter threshold of 3.3 m, a 0.1 error threshold between e_k^2 and d_k^2 for the convergence parameters, a minimum error for $d_k^2 = 0.18$ m, and in the hybrid case a maximum tangent plane error $e_{Tp} = 0.18$ m. Note that the parameters for Helena's data sets are smaller given its finer granularity and smaller noise levels than those in the Smartter data set.

The ICP algorithm only computes relative transformations between consecutively acquired point clouds. To get a view of the fully corrected map (more strictly, of this new augmented odometry, since no loop closure is being performed at this time) it is necessary to concatenate for each pose the corresponding correction.

Next we show comparative results of conventional ICP vs our Hybrid Hierarchic ICP method. Figures 2.10- 2.15 give an empirical comparison between the different methods.

Fig. 2.10(a) shows the MSE error in sq. meters for consecutive paired scans in Helena's dataset once the ICP algorithm has been executed. Frame 2.10(b) shows the



(a) (b)

Figure 2.10: Error comparison on Helena's dataset for each ICP method implemented. Cyan for point-to-point, blue for point-to-plane, green for hybrid with Euclidean distance, and red for hybrid with $l_m = 50$. a) Final positional squared error d_k^2 in sq. meters. b) Final angular error in radians.

final orientation alignment error also for consecutive point clouds. The colored lines indicate the version of the algorithm used: cyan for point-to-point, blue for point-to-plane, green for hybrid with $l_m = \infty$, and red for hybrid with $l_m = 50$. The revised odometric trajectories for the various versions of the algorithm are shown in Figs. 2.11-2.14.

The same is shown for the Smartter dataset. Fig. 2.15(a) shows the MSE error for consecutive ICP pairings and Fig. 2.15(b) shows the error in orientation.

Even when the point to plane metric is more accurate than the other methods for the computation of ICP registration on consecutive point clouds in the case of the Helena dataset, it is unfortunately a very computationally expensive method. The hybrid approach presented is a reasonable trade off between accuracy and computational burden in this case. Interestingly enough, for the case of the Smartter dataset with much denser point clouds, the point-to-point metric performs comparably with the point-to-plane metric, but the hybrid method supersedes the two of them. The reason for this is among other things the adequate sampling of data points and outlier removal.

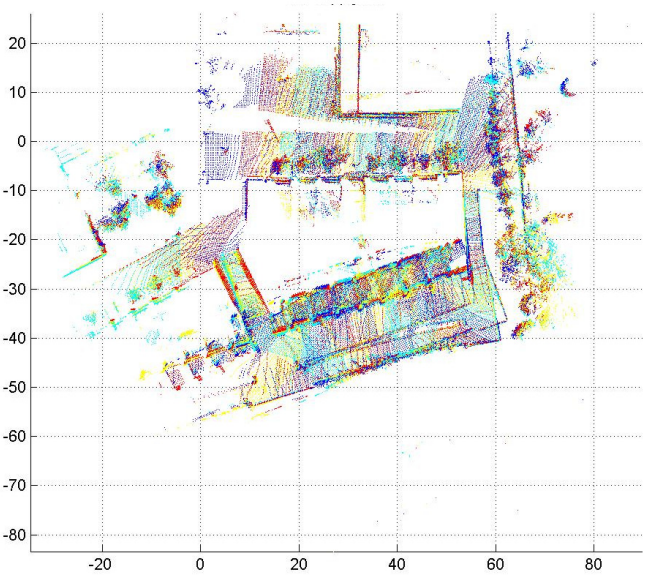


Figure 2.11: Helena dataset, point-to-point metric.

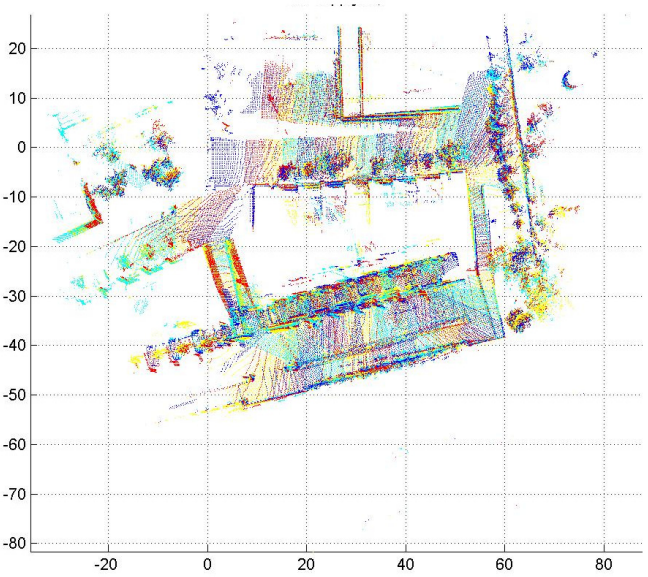


Figure 2.12: Helena dataset, point-to-plane metric.

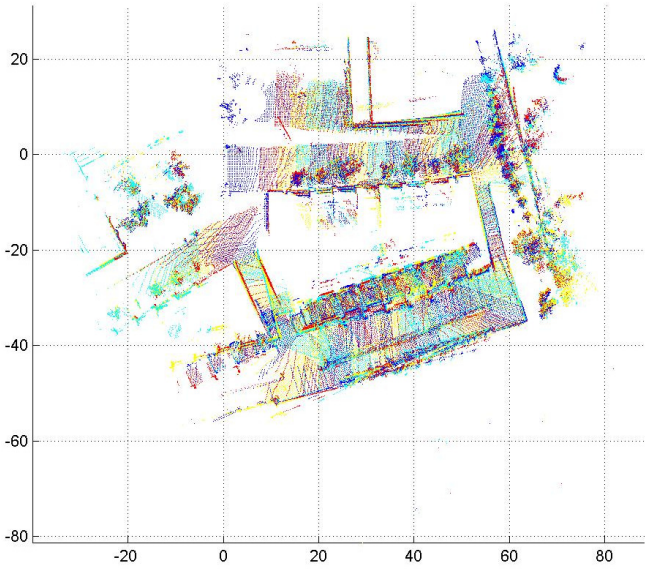


Figure 2.13: Helena dataset, hybrid metric with Euclidean distance.

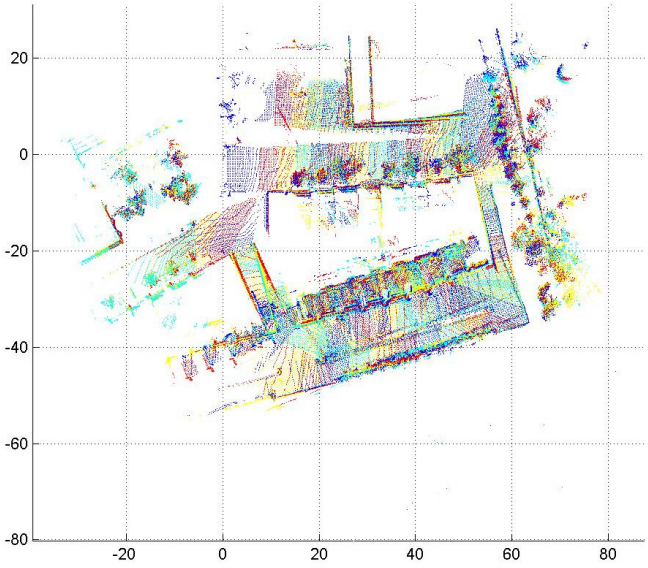
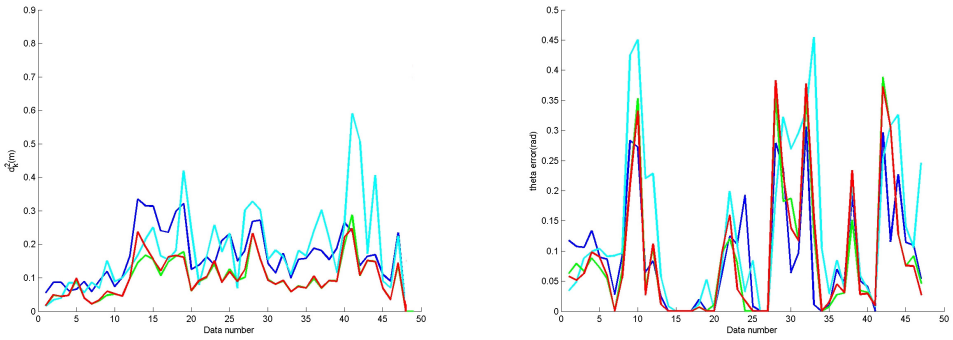


Figure 2.14: Helena dataset, hybrid metric, $l_m = 50$.



(a) (b)

Figure 2.15: Error comparison on Smartter dataset for each ICP method implemented. Cyan for point-to-point, blue for point-to-plane, green for hybrid with Euclidean distance, and red for hybrid with $l_m = 50$. a) Final positional squared error d_k^2 in sq. meters. b) Final angular error in radians.

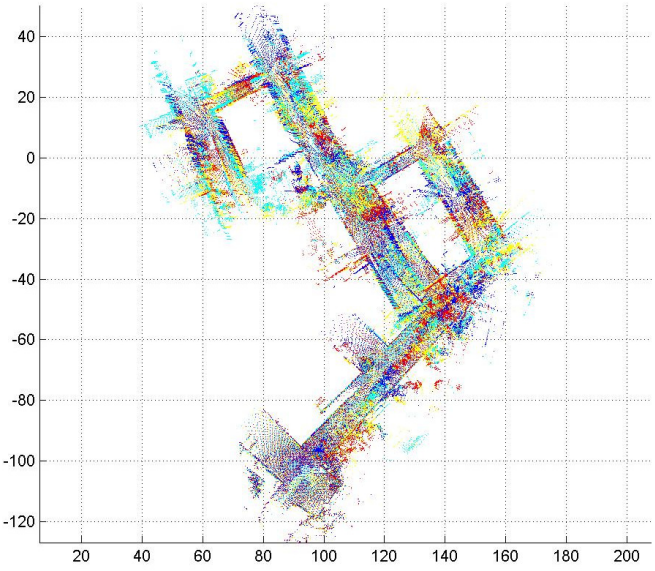


Figure 2.16: Smartter dataset, point-to-point metric.

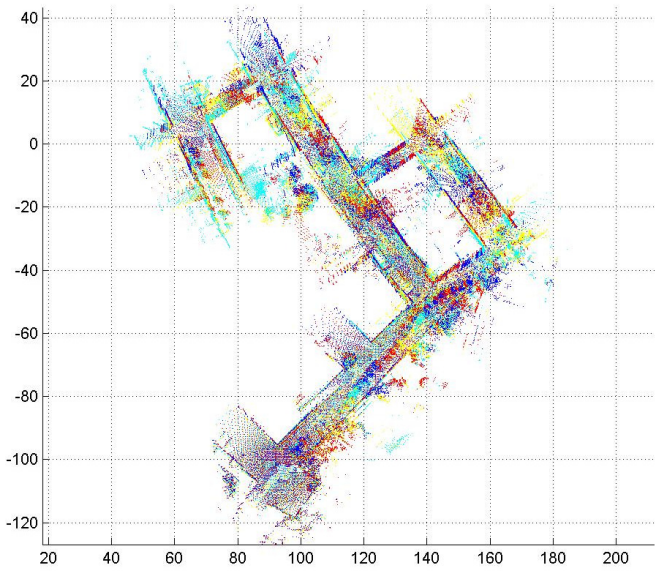


Figure 2.17: Smartter dataset, point-to-plane metric.

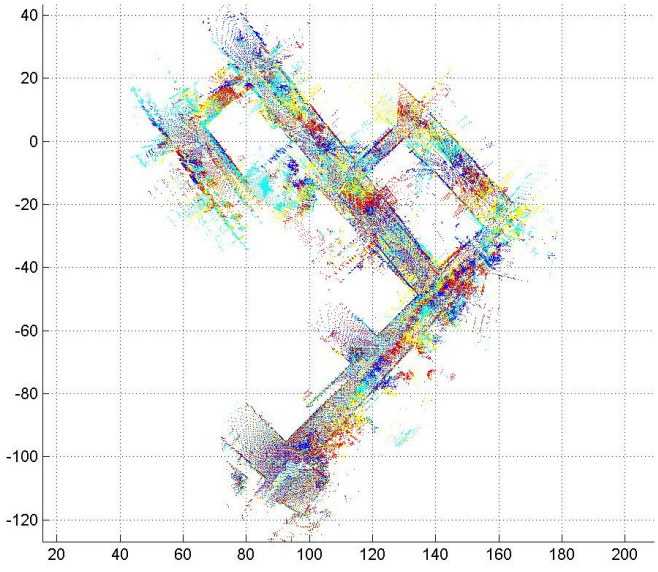


Figure 2.18: Smartter dataser, hybrid metric, Euclidean distance.

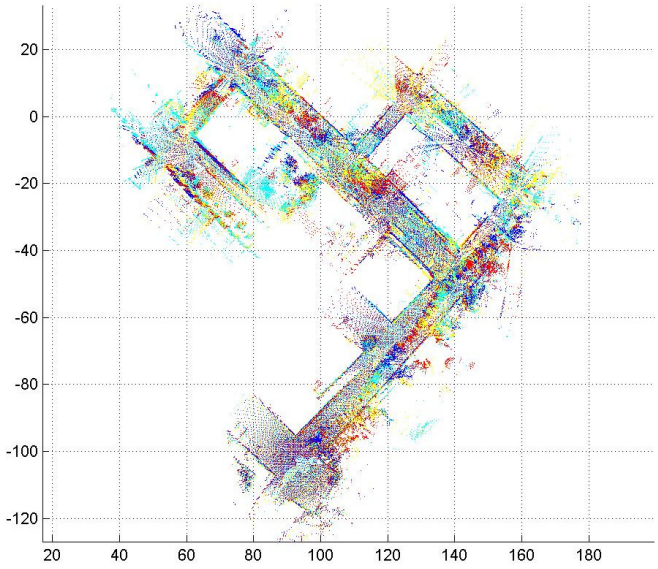


Figure 2.19: Smartter dataset, hybrid metric, $l_m = 50$.

2.4 Fast multi scan alignment with partially known correspondences

Most SLAM algorithms keep probabilistic estimates of the robot location that can be used to determine whether or not a loop closure test is advisable, for instance, by considering not only pose uncertainty but information content as well [51]. But, once a loop closure test is deemed necessary, an algorithm that can compute it expeditiously is needed. Typically loop closure tests are checked not only from the current point cloud to a query point cloud in the past, but instead, to a consecutive set of query point clouds in the past, which in turn have already been registered among them. Using this knowledge, we can expedite multiple registrations at a time. For this reason we proposed FaMSA [129], a technique for fast multi-scan point cloud alignment at loop closure that takes advantage of the asserted point correspondences during sequential scan matching.

Correspondence search is the most expensive step in the ICP algorithm. Finding the NN to a given query point relies on the ability to discard large portions of the data with simple tests. Brute force correspondence search would take $O(n)$ worst case with expected cost $log(n)$, with n the size of the point cloud. The preferred data structures used to solve the NN problem in low multidimensional spaces are kd-trees [37] with $O(n \log n)$ construction complexity and $O(\log n)$ search complexity. They were suggested to use in the ICP in [14], later demonstrated in [149] as an alternative, and finally implemented in [98, 122] to speed up the correspondence search. Another kind of space partitioning are box structures. They take polynomial time to build [2] and constant time to search. However the use of box structures in ICP is only possible when the initial and final poses do not change significantly so that NNs remain in the originally computed box and in the adjacent boxes. We propose to use a modified version of the box structure in [2] to solve the multi-alignment problem.

2.4.1 Using kd-trees for nearest neighbor search

The kd-trees are a generalization of the binary search trees. The idea behind this data structure (trees) is to extend the notion of a one dimension tree on a recursive subdivision of the space, i.e. for the 2D case the subdivision alternates in using the x

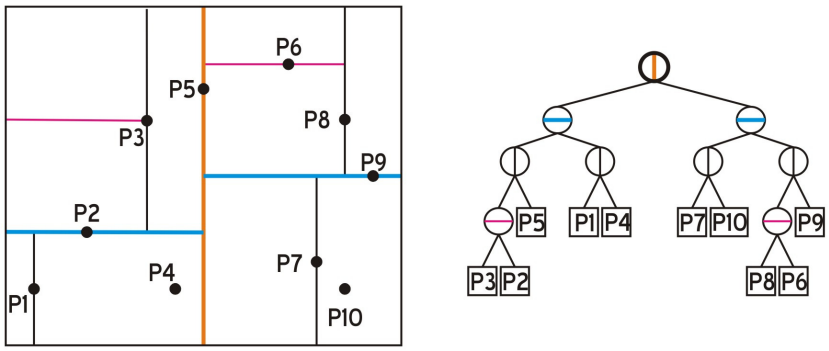
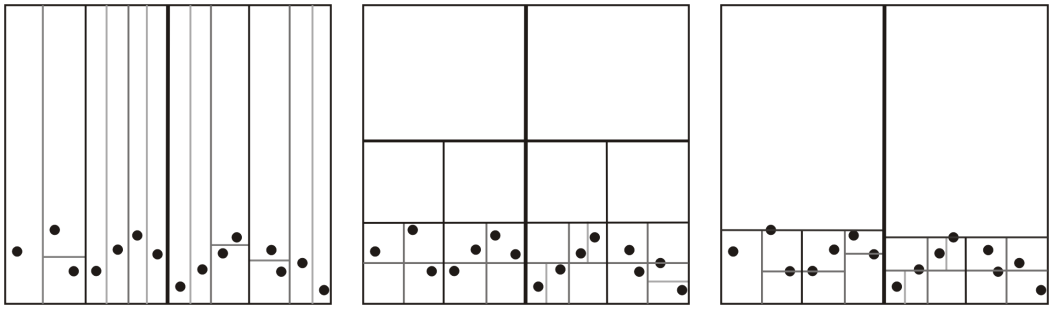


Figure 2.20: A kd-tree example: subdivided data (left) and the corresponding binary tree (right).

or y coordinates to split, Fig. 2.20 (left) . Therefore, we first split on x, next on y, then again on x, and so on. In general, the kd-tree cycles among the various possible splitting dimensions. Each partition (of a point set) is represented by a node containing the two successor nodes or by a bounding box that contains the data points for that node, Fig. 2.20 (right). The root node represents the whole point cloud and the leafs form a disjunct partition of the set. As long as the number of data points associated with a node is greater than a small quantity, called the bucket size (in [37] it is proved that a bucket size of 1 is optimal), the box is split into two boxes by an axis-orthogonal hyperplane that intersects this box.

There are different splitting rules which determine how this hyperplane is selected. The choice of the splitting rule affects the shape of cells and the structure of the resulting tree.

- Standard splitting rule: The splitting dimension is the dimension of the maximum spread (difference between the maximum and minimum values), leading to many cells with high aspect ratio, Fig. 2.21(a). The splitting point is the median of the coordinates along this dimension. A median partition of the points is then performed. This rule guarantees that the final tree has height ($log_2 n$), also guarantees that every kd-tree entry has the same probability. Friedman et al. [37] introduced this splitting rule in their definition of the optimized kd-tree.



(a) (b) (c)

Figure 2.21: kd-tree splitting rules. a) Standard split. b) Midpoint split. c)Sliding-midpoint split

- Midpoint splitting rule: When splitting the space, to guarantee that the tree is balanced, the most common method is the midpoint splitting rule. The splitting value is the median splitting coordinate, Fig. 2.21(b). As a result, the tree will have $O(logn)$ height.

- Sliding-midpoint splitting rule: First a midpoint split is attempted. If the data points lie on both sides of the splitting plane then the splitting plane remains there. However, if all the data points lie to one side of the splitting plane, then the splitting plane "slides" toward the data points until it encounters the first point. One child is a leaf cell containing this single point, and the algorithm recurses on the remaining points, Fig. 2.21(c).

It had been shown that a $O(\log n)$ query time is possible in the average case through the use of kd-trees [37]. Their use ensures that the nearest data point to a query point could be found efficiently. High-dimensional (at least three) NN problems arise naturally when complex objects are represented by vectors of d numeric features.

When using tree structures, finding the nearest neighbor to a given query point relies on the ability to discard large portions of the tree by performing a simple test. The tree is searched in a depth-first fashion and at each stage it makes an approximation to the nearest distance. When the algorithm decides that there cannot possibly be a closer point it terminates, giving the nearest neighbor.

Nearest neighbor search with kd-trees proceed as follows. First, the root node is examined with an initial assumption that the smallest distance to the next point is infinite. The subdomain (right or left), which is a hyper-rectangle (in 3D space this is a rectangular prism), containing the target point is searched. This is done recursively until a final minimum region containing the node is found. The algorithm then (through recursion) examines each parent node, seeing if it is possible for the other domain to contain a point that is closer. This is performed by testing for the possibility of intersection between the hyper-rectangle and the hypersphere (a plain sphere in 3D) formed by target node and distance to the current best NN estimate. If the rectangle that has not been recursively examined yet does not intersect this sphere, then there is no way that the rectangle can contain a point that is a better nearest neighbor. This is repeated until all domains are either searched or discarded, thus leaving the nearest neighbor as the final result. In addition the algorithm not only provides the NN, but also the square of the distance to the NN. Finding the nearest point is an $O(logN)$ operation.

For the kd-tree implementation (like in [95]) we use the Approximate Nearest Neighbor (ANN) library by [8]. ANN is a library of C++ objects and procedures that supports the NN search and the approximate nearest neighbor search. It is designed for data sets that can be stored in main memory. Points are assumed to be represented as coordinate vectors of reals. The distance between two points can be defined in many ways. ANN assumes that distances are measured using any class of distance functions called Minkowski metrics, including the Euclidean distance, Manhattan distance, and max distance. Preprocessing time and space are both linear in the number of points n and the dimension d. Thus the data structure requires storage that is only moderately larger than the underlying data set. Also it supports kd-trees [13, 37], and box-decomposition trees [8], it is able to use different methods for building these search structures and it also supports two methods for searching these structures: standard tree-ordered search [9] and priority search [8].

2.4.2 Box structures for nearest neighbor search and multi-scan matching

The box structure is a very simple space representation where after partitioning the space, a point's correspondence is searched only in the box containing this element and in the adjacent boxes. The idea is, for a given point cloud P_1 representing spatial information, a list $\boldsymbol{I}_1$ is maintained in spatially non-ordered form. The boxing data structure is built with a rearranged $\boldsymbol{I}_1$ (of size n) as list $\boldsymbol{I}_2$ and an index matrix $\boldsymbol{I} = \boldsymbol{I}_{u,v,w}$ whose elements are associated to individual boxes: $u, v, w = 0, ..., m-1$, and contains integers indicating the beginning of the boxes in $\boldsymbol{I}_2$, where m is the number of boxes along the axis.

The memory requirement is of order $O(n)$ for $\boldsymbol{I}_2$ and $O(m^3)$ for $\boldsymbol{I}$. For the sake of clarity of the explanation, $\boldsymbol{I}_2$ is given as a point list containing the (x, y, z) coordinate values. To keep $\boldsymbol{I}_1$ in memory, then $\boldsymbol{I}_2$ should only contain the access index to $\boldsymbol{I}_1$ or pointers which directly point to the memory locations of the point coordinates.

The box structure limits are given by the maximum and minimum points of the involved point clouds. The access procedure requires $O(k)$ operations, where k is the average number of points in the box. One of the main advantages of the boxing structure is a faster and easier access mechanism than that of the tree search-based methods. Since for the query point we need only to compute the corresponding box and its adjacent one.

Also there is no need neither to re-establish the boxing structure nor to update the $\boldsymbol{I}$ and $\boldsymbol{I}_2$ lists in each iteration. Only four positions are updated in the course of iterations, Fig. 2.22. They uniquely define the boxing structure under the similarity transformation.

In their implementation [2], the correspondence is searched in the boxing structure during the first few iterations, and in the meantime its evolution is tracked across the iterations. Afterwards, the searching process is carried out only in an adaptive local neighborhood, according to the previous position and change of correspondence. In any step of the iteration, if the change of correspondence for a surface element exceeds a limit value, or oscillates, the search procedure for this element is returned to the boxing structure again.

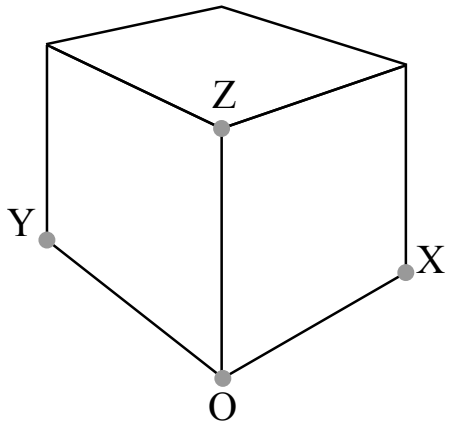


Figure 2.22: The boxing structure bounds all the data points inside, and requires the update of only four positions at each iteration.

Differently from [2], where they assign to empty boxes the index of the last occupied box, we instead leave empty boxes out of the search. This serves effectively as a fixed distance filter with significant savings in computational load. The proposed multi-alignment method, is faster than using the optimized Approximate Nearest Neighbor (ANN) library [8] with fixed radius search, as shown in the experiments section.

We have already mentioned that correspondence search is the most expensive part of any ICP implementation, for this reason we propose FaMSA to boost multiple scan alignment using previously known correspondences. That is, given two previously aligned point clouds P and P', the relative transformation (x) between them, and a list Y of correspondences, we want to find the registration between the current query point cloud Q and the two different scans P and P'.

The method proceeds as follows. Standard correspondence search is implemented between clouds P and Q, and for each match between points p_i and $\mathbf{q}_i$, a link to P' is read from Y, and consequently the distance from $\mathbf{q}_j$ to p'_k is immediately established, avoiding the computation of similarity search and filters. Aside from the previous alignment of P and P', the method needs, as any other iterative ICP algorithm, an initial estimate of the relative displacement between the query cloud Q and P. Algorithm 2 shows the approach.

In the algorithm, Z and Z' indicate the correspondence sets between P and Q; and

FaMSA$(P, P', Q, Y, \mathrm{x}, \mathrm{x}_0)$

Inputs:

P, P':	Two consecutive query point clouds.
Q:	Current point cloud.
Y:	Correspondences between P and P'.
x:	Relative displacement between P and P'.
x_0:	Initial displacement between P and Q.

Outputs:

x_P:	Relative displacement between P and Q.
$\mathrm{x}_{P'}$:	Relative displacement between P' and Q.

1: $\mathrm{x}_P \leftarrow \mathrm{x}_0$
2: $\mathrm{x}_{P'} \leftarrow (\mathrm{x}_0 \oplus \mathrm{x})$
3: **while** not convergence **do**
4: $\quad Z \leftarrow$ NNSearch(P, Q, R_P, t_P)
5: $\quad Z' \leftarrow$ Link(Z, Y)
6: $\quad \mathrm{x}_P \leftarrow$ ICPUpdate(P, Q, x_P, Z)
7: $\quad \mathrm{x}_{P'} \leftarrow$ ICPUpdate$(P', Q, \mathrm{x}_{P'}, Z')$
8: $\quad$ convergence $\leftarrow (\epsilon < T)$ **and** $(\epsilon' < T)$
9: **end while**

Algorithm 2: FaMSA: Fast multi-scan alignment with partial known correspondences.

P' and Q, respectively. Appropriate index bookkeeping links to the other in constant time. The threshold T is used to indicate the maximum error allowed for the registration of both point clouds. The method also limits the search to a maximum number of iterations, typically set to 100.

The method is suboptimal in the sense that no new matches are sought for between point clouds P' and Q. For sufficiently close reference clouds P and P' it does not impose a limitation on the quality of the final correspondence.

In the same way that FaMSA takes advantage of the point correspondences between P and P' to boost the computation of the relative displacement between P' and Q, one can also defer the estimation of the pose between P' and Q until all iterations for P have finished and use the result as a starting point for the second optimization. This method is shown in Algorithm 3.

Extensive experimentation shows that only one iteration of ICP update suffices to revise the pose of P' with respect to Q, once the relative transformation between P and Q has been optimized. We call this method FaMSA2.

FaMSA2($P, P', Q, Y, \mathrm{x}, \mathrm{x}_0$)
Inputs:
P, P': Two consecutive query point clouds.
Q: Current point cloud.
Y: Correspondences between P and P'.
x: Relative displacement between P and P'.
x_0: Initial displacement between P and Q.
Outputs:
x_P: Relative displacement between P and Q.
$\mathrm{x}_{P'}$: Relative displacement between P' and Q.

Require:
$\mathrm{x}_P \leftarrow \mathrm{x}_0$
while not convergence **do**
  $Z \leftarrow$ NNSearch(P, Q, x_P)
  $\mathrm{x}_P \leftarrow$ ICPUpdate(P, Q, x_P, Z)
  convergence $\leftarrow (\epsilon < T)$
end while
$\mathrm{x}_{P'} \leftarrow (R_P, t_P) \oplus (R, t)$
while not convergence **do**
  $Z' \leftarrow$ Link(Z, Y)
  $\mathrm{x}_{P'} \leftarrow$ ICPUpdate$(P', Q', \mathrm{x}_{P'}, Z')$
end while

Algorithm 3: FaMSA2: Very fast multi-scan alignment with partial known correspondences.

2.4.3 Barcelona Robot Lab

During the course of our research, the URUS project was evolving and we needed to map larger areas than those covered by the two datasets used in the previous section. To that end we produced a more complete and much larger dataset. This dataset was accompanied with data coming from other sensors such as cameras and odometry priors. At the time this research was developed, there were little to nonexistent databases with enough data and detail to thoroughly test our methods, so we had to create one ourselves.

This dataset [131] was gathered also at the Barcelona Robot Lab (BRL) area mentioned in the previous chapter, an urban like setting placed at Campus Nord of the Universitat Politècnica de Catalunya (UPC). Fig. 2.23 shows the physical distribution of the camera network in the Barcelona Robot Lab. And Fig. 2.24 shows imagery from a pair of these during the dataset acquisition session along with a sample captured scan.

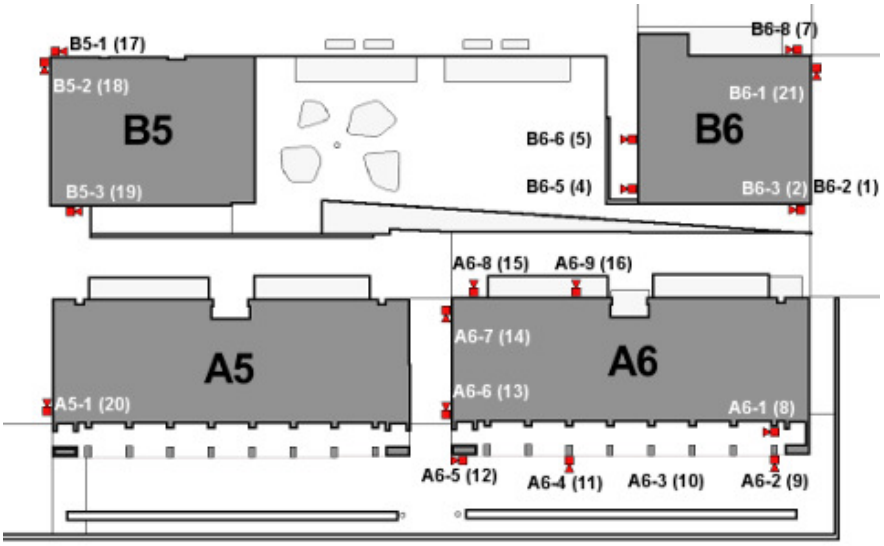


Figure 2.23: Camera network distribution at Barcelona Robot Lab.

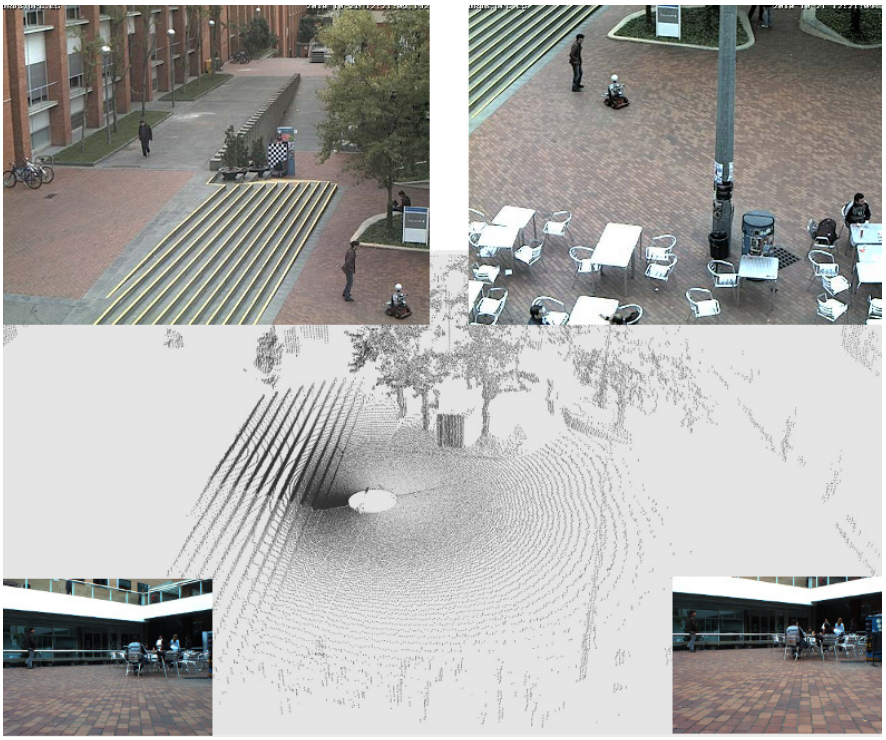
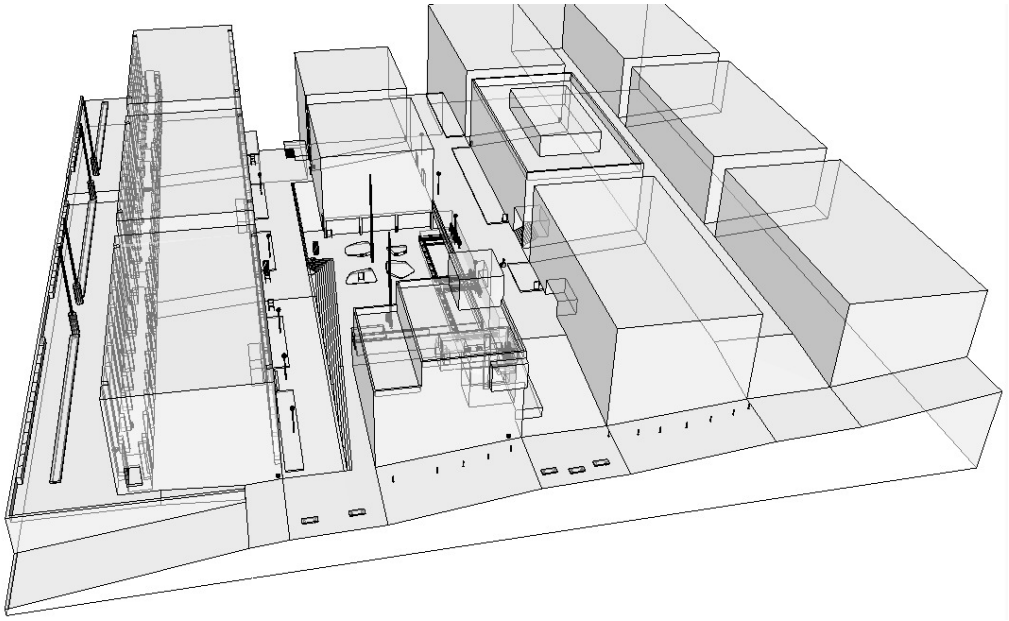
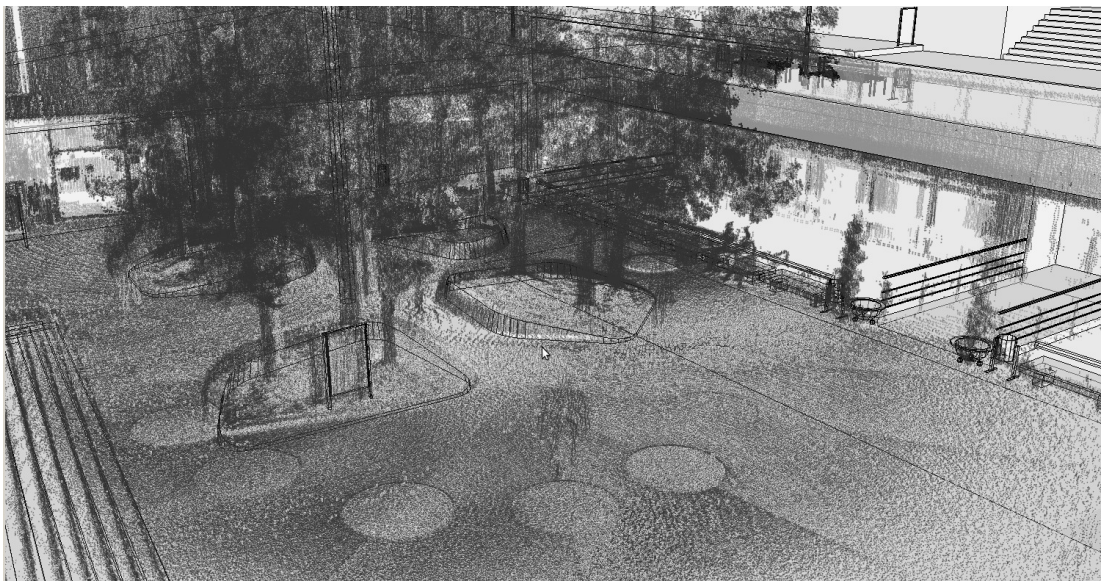


Figure 2.24: Two BRL cameras (up) in the plaza and corresponding 3D range scan (center) with stereo images (down).



(a)



(b)

Figure 2.25: a) Virtual model of the BRL scenario. b) Overimposed 3D cloud in the virtual model.

The data set is intended but not limited to benchmarking algorithms for robust outdoor localization, mapping and navigation, scene understanding, path planning and motion pattern analysis in both the robotics and computer vision communities.

It contains over 12GB of dataand surveys an area that covers 10,000 square meters, it includes 3D point clouds of the whole area, on-board robot imagery, as well as imagery from a camera sensor network. Data acquisition was spanned in two consecutive days.

Three aspects made this dataset unique at the time our research was taking place:

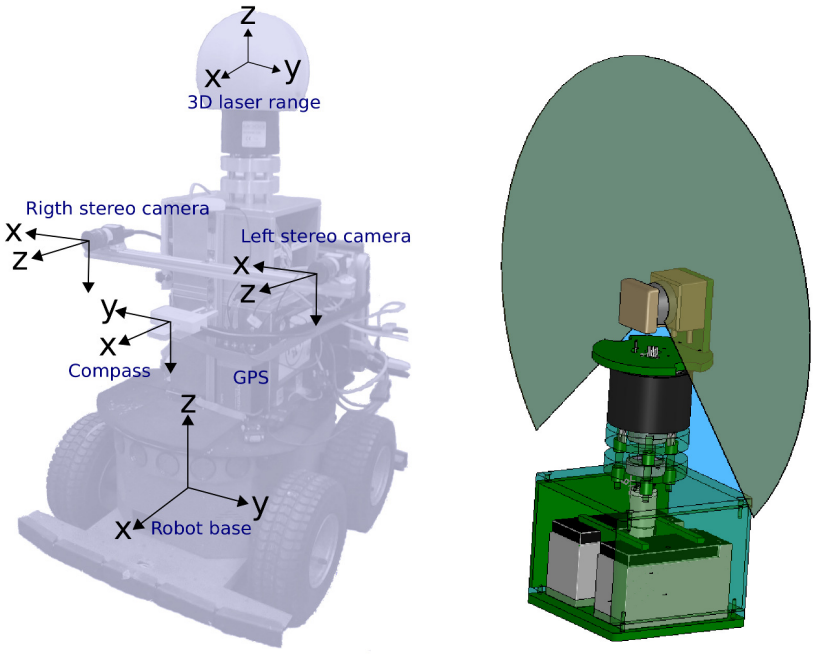
1. It includes a time-stamped sequence of images from the entire camera sensor network spanning the two days of data gathering.

2. It contains impressively rich 3D point clouds obtained with our proprietary built 3D scanner.

3. We provide a 3D virtual model of the scenario for ground truth comparison, *i.e* a obj file, see Fig. 2.25.

The data set contains high quality 3D range data from a proprietary 3D laser, stereo images, compass and the camera network images, along with their calibration parameters, and a 3D virtual model of the scenario. All data has been carefully timestamped with logs in a human readable format. The dataset spans two consecutive days, leading to 400 3D range scans (2.7 Gb), 800 stereo images at high resolution (5.5 Gb), and more than 25,000 images from the camera network (3.7Gb). We also provide parsing tools and examples written in MATLAB to access the data easily. The dataset is freely available at the web site [131] .

Aside from the sensor data, it includes a 3D CAD model of the BRL area, see Fig. 2.25. The map was built by hand, taking measurements with laser distance meters and measuring tape, delivering a coherent 3D model which incorporates the most important geometrical elements such as buildings, stairs, ramps, borders, curbs, some vegetation elements and urban furniture such as benches or streetlights. The 3D model is provided in *.obj* geometry definition file format [133], originally developed for 3D computer animation and scene description, which has become an open format and a *de facto* exchange standard, see [24] for further details. This model was used successfully by 3D localization algorithms [136].

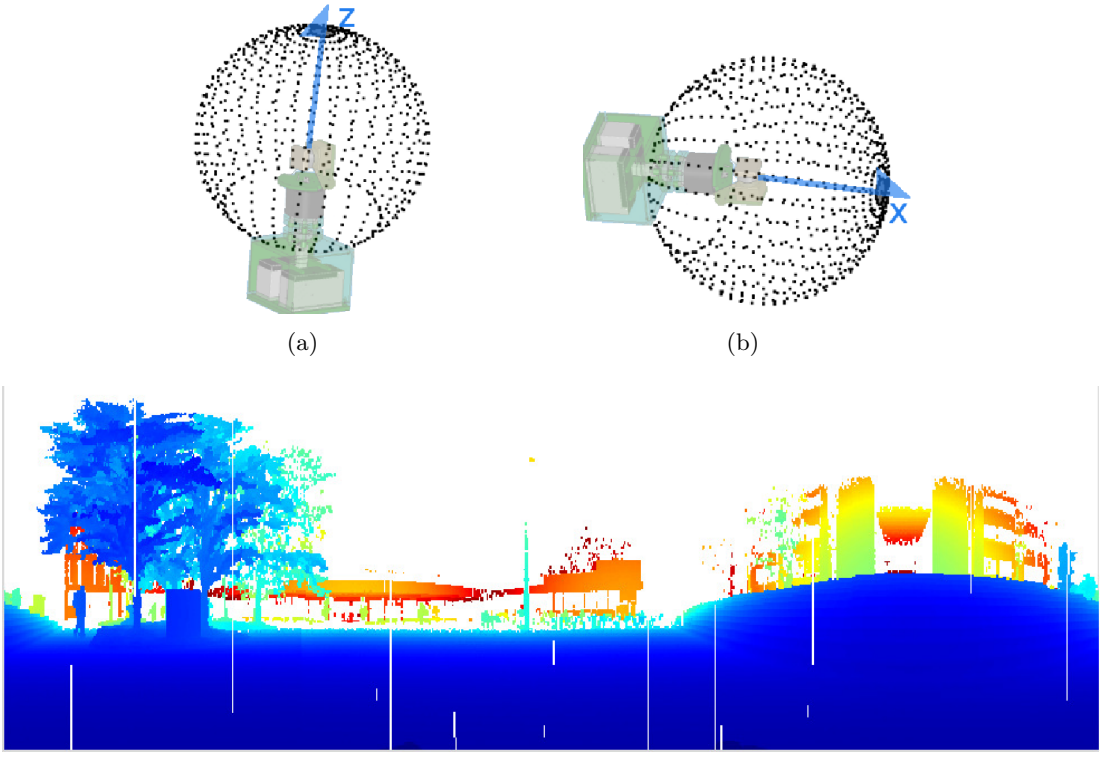


(a) (b)

Figure 2.26: a) Our robot Helena with the second custom made 3D laser and the sensors reference frames. b) Our custom 3D laser based on a Hokuyo-UTM30 2D range finder.

The robotic platform used to gather the dataset was a Pioneer3 AT mobile robot, called "Helena", see Fig. 2.26(a). Helena had a Centrino at 1.6GHz laptop for control and to record proprioception sensors, such as odometry and IMU. Additionally, Helena had a dual-core 2GHz laptop to record the 3D range scanner and images from the stereo cameras.

As mentioned earlier, in the process of this thesis, we built two different laser range scanning systems. The first custom 3D sensor we built was a tilting unit, with a RS4-Leuze 2D laser [87]. It had the disadvantage of only scanning when the mechanism was tilting up, and was mechanically unstable and difficult to control. Moreover, it had a limited field of view horizontally and vertically. Consequently, for the BRL dataset, we decided to build a second 3D range sensor. We opted to mount a more compact 2D range sensor in a slip-ring, allowing the sensor to be always rotating, thus avoiding the dead time from the tilting unit, and warranting continuous scans. Moreover, as



(a) (b)

(c)

Figure 2.27: Proprietary 3D range laser, Hokuyo UTM-30LX based. Example of the laser scans a) in yaw top configuration, b) in roll front configuration. c) Range image acquired with the 3D laser in yaw top configuration. Blue means closer range and red farthest.

the laser was mounted atop the motor, it was mechanically more stable and easier to control compared with the tilting unit. We employed a Hokuyo UTM-30LX scanner (30 meters range), were the angular position is controlled with a DC brush-less motor and a computer. The system azimuth (horizontal) resolution is 0.5° over (360°) and zenith (vertical) resolution is 0.25° (over 270°) leading to maximum definition up to 194,580 points per scan, with 30 meters of range and radial error about 30 mm for the interval of 0.1 to 10 m, and 50 mm for 10 to 30 m.

Helena had the ability to explore indoors and outdoors environments, it was able to climb hills on flat floor up to 30° and carry payloads up to 12 kg, including additional batteries. Base autonomy was 3-6 hours on three fully 12V 7Ah charged batteries.

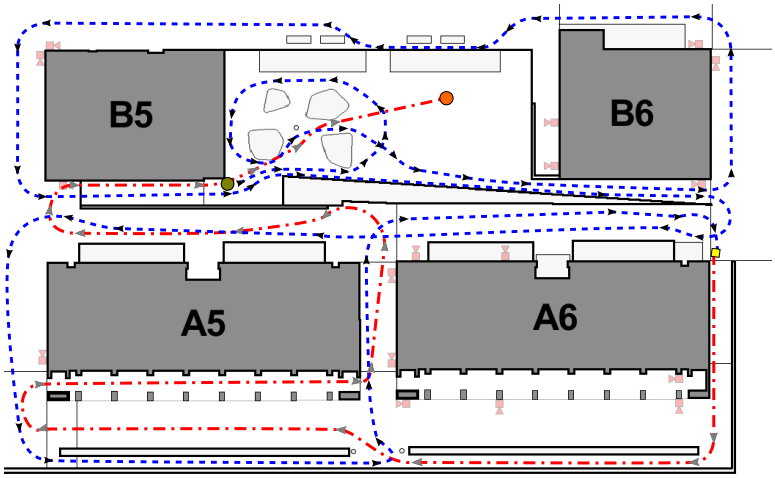


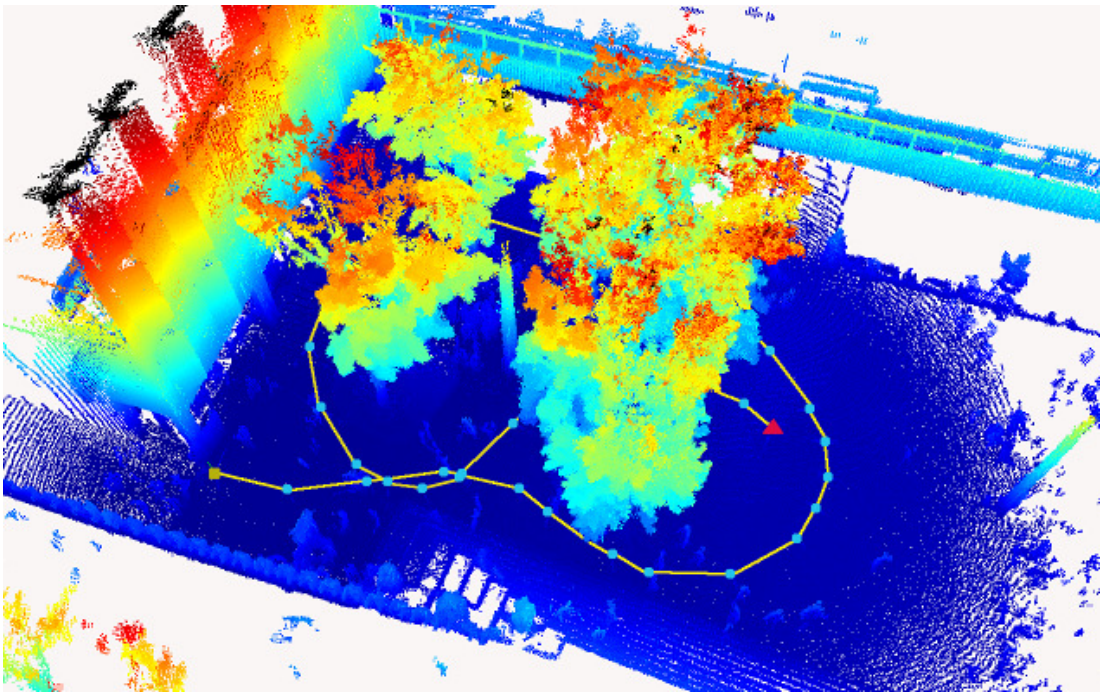
Figure 2.28: Data set workspace in the BRL, with the robot trajectory for both days. The first day is the blue dotted line and the second day is the red line.

Our 3D range scanner was designed to allow for two scanning configurations, either in yaw top configuration, or in roll front configuration, Fig. 2.27. The BRL dataset was acquired with a yaw top configuration. The sensor is mounted in top of our robotic platform to minimize possible occlusion from other sensors, or the vehicle itself. Helena also carries a proprietary stereo camera system based on two FLEA 2 cameras with a 418 mm baseline and COMPUTAR M1214-MP lenses, with a field of view of 40.4°. The cameras offer a resolution of 1280x960. Our robot delivers odometry data in a (x,y,θ) format, and carries a TCM3 compass to obtain the orientation about the 3 axis.

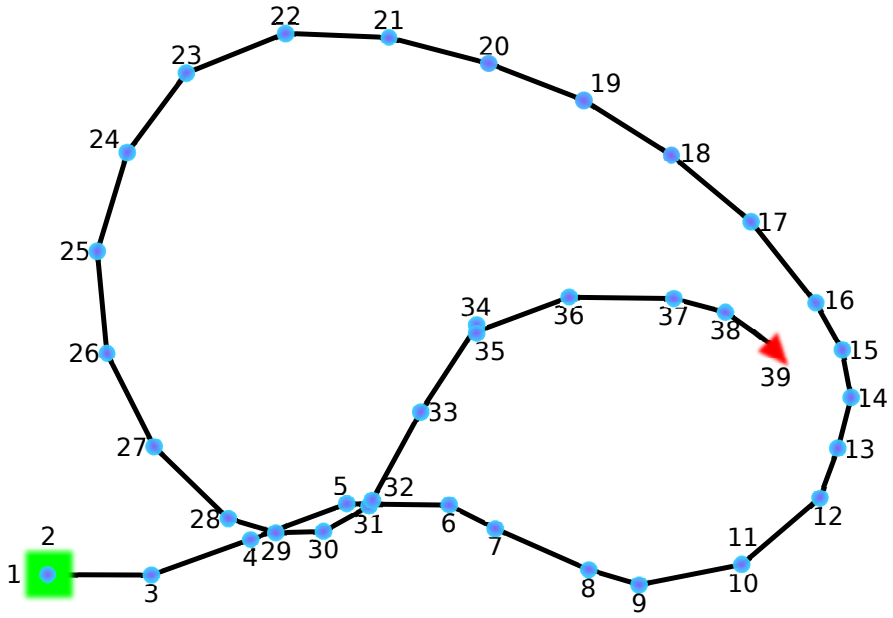
To gather the data it was necessary to expend two days of experiments. Figure 2.28 shows the trajectory done by Helena at the BRL area during the two spanned days.

2.4.4 FAMSA execution times

We now report results of our fast scan alignment algorithm with partially known correspondences on this dataset. For this experiment we only use the first 39 point clouds of the BRL dataset. Fig. 2.29 shows a view of the reconstructed part of the environment. Each scan was uniformly sampled for faster convergence using voxel space discretization with a voxel size of 0.35 meters. During sampling, we also computed surface normals and enforced a minimum voxel occupancy restriction of 4 points. Random sampling

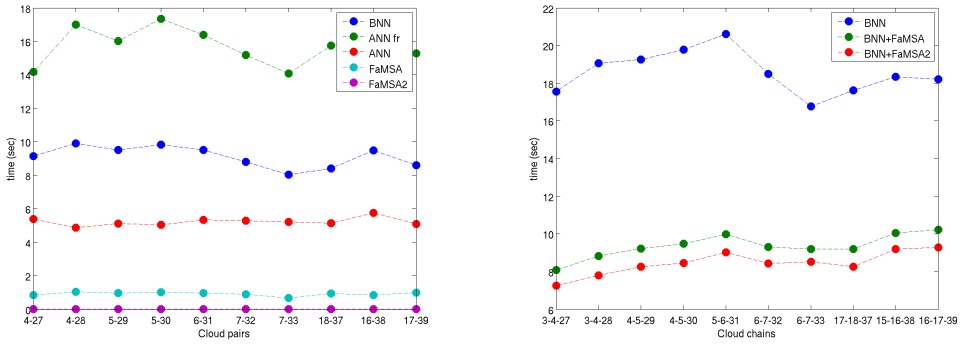


(a)



(b)

Figure 2.29: A path with 39 poses around the FIB plaza of the Barcelona Robot Lab. a) Dense point cloud registration, color indicates height. b) Robot trajectory, in green the initial pose, in red the final pose.



(a) (b)

Figure 2.30: Algorithm performance. a) Time required to match Q and P', when the correspondences between P and P' are known. b) Time required to match Q with both P and P'.

with set sizes of 20 points was used for those boxes exceeding such number of points. Normal orientations are computed after random sampling. This has shown to produced better orientation estimates, especially around corners, when compared to other strategies such as k-NNs with density filtering.

ICP is executed pairwise and in open loop for the 39 consecutive scans, storing all relative pose displacements as well as the correspondence indexes. Then, a number of possible loop closure locations were selected manually. FaMSA was executed on these loop closure candidates. The specific parameters of the ICP implementation include: maximum angle between normals of 35 deg; upper and lower bounds of sigma rejection at 0.25σ and 5σ, respectively; and maximum number of iterations at 100.

For the execution times reported, experiments were run in MATLAB using mex files of C++ routines in an Intel Core 2 Quad CPU Q9650 3.0 GHz system, with 4 GB RAM running Ubuntu 10.04 32 bits.

We first compare the execution time in seconds for various implementations of multi-scan ICP. To this end, 10 loop closure locations Q are selected in the trajectory, and each is compared against its query clouds P and P'. Assuming that an alignment between a cloud in Q and another one in P has been established, we compute the time it takes to align the current cloud Q to the second query cloud P' given the correspondences between Q and P are known, see Fig. 2.30(a). The methods BNN, ANN-FR and

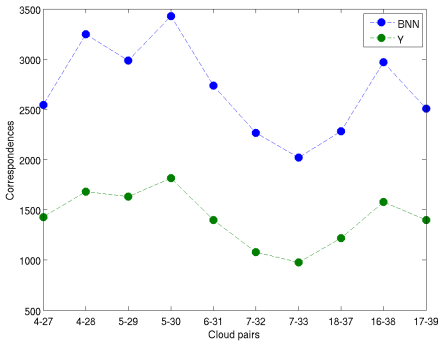


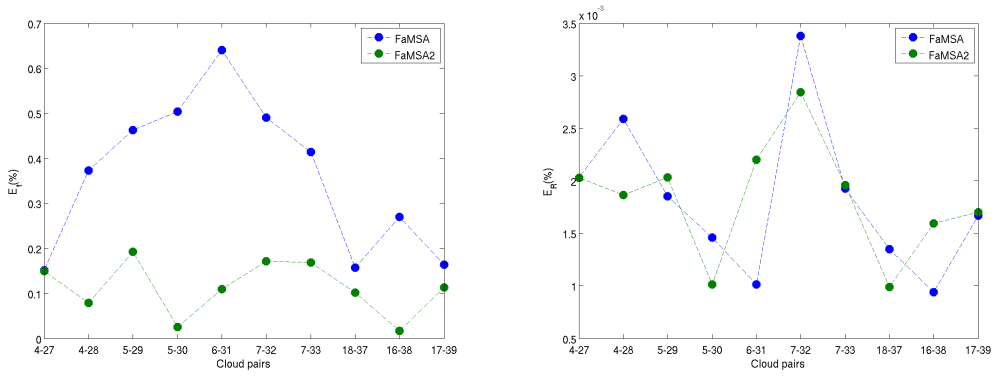
Figure 2.31: Number of correspondences between P' and Q running a full BNN compared to using the stored set Y.

ANN refer to our implementation of voxel NNs; ANN with fixed radius, the size of the voxels; and conventional ANN. FaMSA and FaMSA2 stand for the methods presented in this chapter that make use of previous point correspondence indexes to speed up registration. Note that FaMSA2 is the fastest of the methods, requiring only one iteration in the minimization. Extensive experimentation showed that further refinement in the case of FaMSA2 does not significantly improve the registration.

Figure 2.30(b) plots the time it takes to register the current point cloud Q against both query clouds P and P'. The plot shows the individual registration using BNN and the combined registration using the proposed schemes BNN+FaMSA and BNN+FaMSA2. Note how the advantages in computational load of using any of the proposed algorithms are significant.

One might think that using only the correspondences in Y (the set of correspondences from clouds P and P') would yield suboptimal estimation. As a matter of fact, when using only this set to compute the relative displacement between P' and Q, the number of correspondences effectively halves (see Fig. 2.31), but pose estimation accuracy does not suffer significantly.

Figure 2.32 plots proportional translation and rotational errors as compared with full ICP estimation using BNN, and computed as follows [86]: using as ground truth the relative pose between Q and P' as computed with BNN (R_{BNN}, t_{BNN}), we measure the relative error of the estimated rotation $\mathbf{R}$, as $E_R(\%) = \|\mathbf{q}_{BNN} - \mathbf{q}\|/\|\mathbf{q}\|$, where

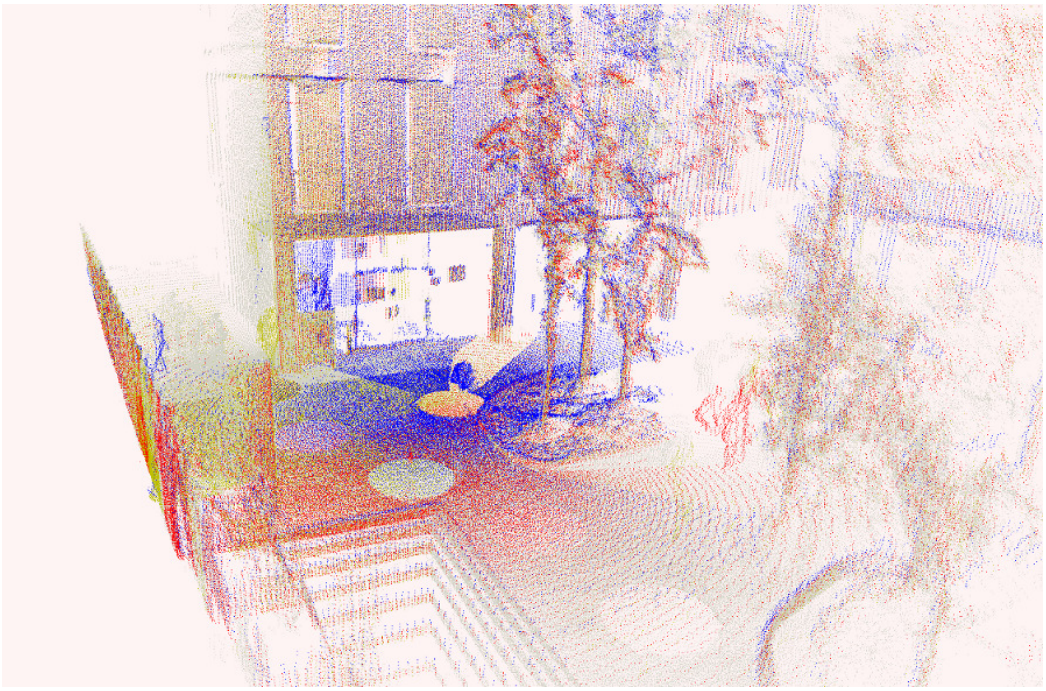


(a) (b)

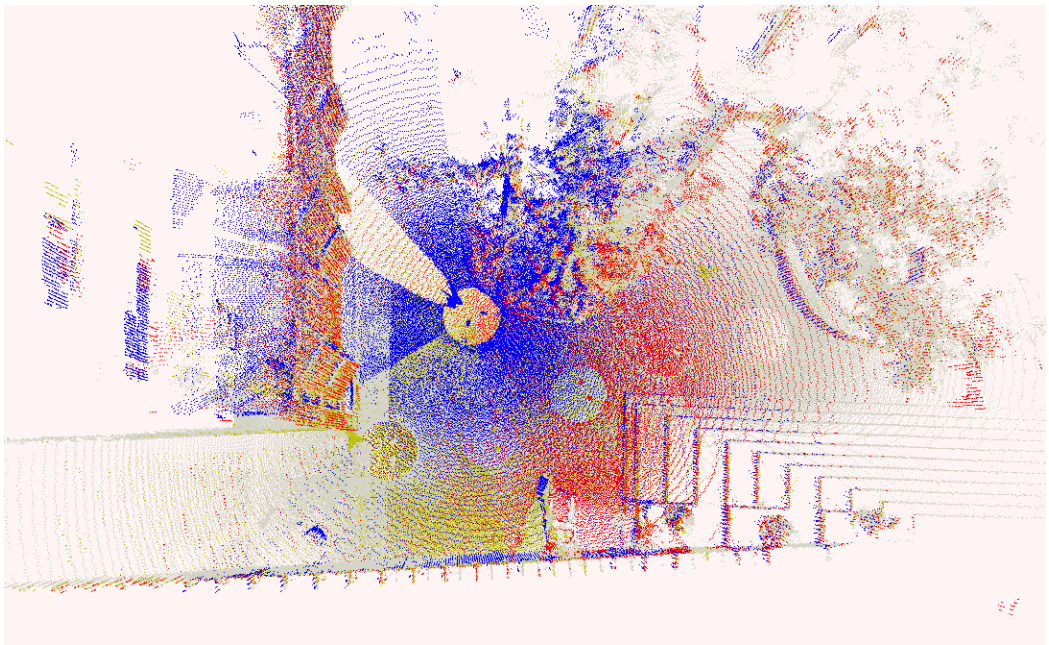
Figure 2.32: Proportional relative error. a) Translation and b) rotational errors for the registration between Q and P' with the proposed methods. BNN is used for ground truth comparison.

$\mathbf{q}_{BNN}$ and $\mathbf{q}$ are the normalized quaternions of the corresponding orientation matrices $\mathbf{R}_{BNN}$ and $\mathbf{R}$, respectively. Similarly, the relative error of the estimated translation is computed with $E_t(\%) = \|\mathbf{t}_{BNN} - \mathbf{t}_{P'}\|/\|\mathbf{t}_{P'}\|$. Translation error turns out to be less than 0.7% for FaMSA and for all cloud pairs, and less than 0.2% for FaMSA2. Rotation error is barely noticeable for both methods.

Figure 2.33 shows a sample of the point cloud match (best viewed in color). In blue, the current pose. In green and red, the query poses. A safe percentage of point cloud overlap in our method is roughly 50%. This is achieved with displacements of about 4 meters.



(a)



(b)

Figure 2.33: A loop closure location between clouds 3, 4, and 28 in the BRL dataset (best viewed in color). a) Perspective and b) top views. P in yellow P' in red, and Q in blue.

Chapter 3

3D range mapping

In this Chapter we present the tools needed to create a globally consistent representation of a robot's environment, which is crucial for localization and navigation. Many mobile robotic systems gather information about their local environment using laser range data [17, 77, 97, 98, 135]. These local representations have to be matched to build a global map. The iterative procedures of local pairwise matching explained in the previous chapter is only part of the picture. Pairwise matching alone leads inevitably to inconsistencies due to the accumulation of small errors in the matching process. To attain a globally consistent map, global matching algorithms are needed, taking if not all, as many possible correspondences between all scans into account.

The above mentioned errors of cumulative pairwise matching could be produced by wrong convergence of the matching algorithm, under-constrained situations, or sensor noise. The first of these factors was addressed in the previous chapter with our hybrid point-to-point and point-to-plane heuristics for correspondence search. The second factor, was addressed also in that chapter by the subsampling strategy. Sensor noise bounds is typically given by the sensor manufacturer. We are thus interested in modeling how such sensor noise propagates through the scan matching algorithm to produce an error estimate of the relative transformation between two point clouds. Modeling this error is important as it allows us to compute motion estimates with the appropriate uncertainty bounds. In this Chapter we present the mathematical tools to estimate the covariance of the relative motion estimate computed by the ICP registration algorithm from chapter 2. These motion estimates, and probability distributions, serve as input to the global mapping algorithm,*i.e* the SLAM method we employ in this thesis. The

motion estimates are used either as odometry measurements when matching consecutive point clouds from consecutive poses in time, or as loop closure constraints, when computing the relative motion of the most recent robot pose with respect to any other previous pose besides the immediately previous one. We also explain in this chapter the estimation machinery behind the chosen SLAM method.

The rest of this Chapter is structured as follows. We start with the related work in Sec. 3.1. Next, we continue with the error propagation model for the presented scan matching Sec. 3.2. Later, in Sec. 3.3 we explain the chosen filtering scheme used for SLAM. And in the last section, Sec. 3.4, we show some of the 3D maps built.

3.1 Related work

Mapping with 3D laser range finders has been addressed in many ways. There exist non-probabilistic approaches, such as the one proposed in [97], where the alignment of two scans is achieved mainly by improvements to the basic ICP algorithm [14]. However, more common methods for merging all scans are based on probabilistic information. Probabilistic methods allow for a straightforward way to distribute errors when closing a loop. That is, when revisiting a location after a long traverse. One possibility for 3D probabilistic SLAM in outdoor environments is to employ a delayed-state framework with an Extended Kalman Filter (EKF) [23].

Using an Extended Information Filter (EIF) within the delayed-sate paradigm has better scalability properties compared to the EKF [32]. A delayed-sate EIF generates exact sparse information matrices and, during open loop traverse, the information matrix becomes tri-block diagonal as consecutive robot poses are added to the state. At loop closure, the matrix structure is only modified sparsely, setting information links between non-consecutive robot poses. Thus, one advantage of the delayed-state information-form for SLAM is that predictions and updates take constant time, assuming an efficient or approximate way for state recovery is used to evaluate Jacobians.

The approximations performed by linearizations, together with covariance and state recovery and data association are issues of concern in the use of EIF filters for SLAM. Our group has proposed an alternative to reduce the overconfidence effects of linearizations by closing only the most informative loops, decreasing the total number of loop

closure links, maintaining the sparsity of the information matrix [50]. The technique not only defers filter inconsistency but also has better scalability properties. As for state recovery in information form, efficient techniques for exact recovery of covariance and state estimates are proposed in [48, 55]. Our group showed later that during open loop traverse, exact state recovery can be performed in constant time [52].

Pose SLAM is one such variant of delayed state SLAM where only the robot trajectory is estimated and in which landmarks are solely used to compute relative constraints between robot poses. We have developed efficient methods to build 2D Pose SLAM maps [51, 52] and 3D Pose SLAM maps [140] that ponder the information content on odometry and measurement links to keep the graph of poses sparse. In this chapter we show results of such Pose SLAM mapping achieved with our custom built 3D lasers.

3.2 Propagation of error from point cloud registration to relative pose estimation

Next, we present a method to model the uncertainty of the relative motion parameters computed with the registration approach just described in Chapter 2. This method propagates the noise from the matching process of a point cloud pair, to that of the motion estimate, thus delivering a probability distribution about the uncertainty in the parameters of the relative transformation.

One way to compute an accurate covariance approximation is by Monte Carlo simulation. However, this is a time-consuming solution, and we prefer to devise a closed-form solution. In [19], the authors provide an analysis of different methods to compute the propagation of the ensuing covariances of the scan matching process; such as, the Hessian method [11, 15], which in some cases greatly over-estimates the real covariance; or first order error propagation, a very well know method for covariance approximation [33], which is our preferred choice. We choose first order approximation because for the case of very small relative motions, nonlinearities play a minor role and the over-approximation is bearable. This first order approximation has also been studied for the case of 2D scan matching [19], and in our institute, for the case of point 3D model acquisition [35] and for camera calibration [38].

The general rule for a first order approximation of the covariance $\Sigma_{\mathbf{x}}$ of the solution $\mathbf{x}$ in Eq. 2.16 would be to compute a first order covariance propagation of the uncertainties in 3D point estimation $\Sigma_{P,P'}$, assuming that Eq. 2.16 is a differentiable function of the point sets (P, P')

$$\Sigma_{\mathbf{x}} = \nabla\mathbf{f}\Sigma_{P,P'}\nabla\mathbf{f}^{\top} \tag{3.1}$$

with $\nabla\mathbf{f}$ the Jacobian of Eq. 2.16 with respect to the 3D correspondences in the point estimates P and P', and

$$\Sigma_{P,P'} = diag\left(\Sigma_P^{\mathsf{p}_1}, ..., \Sigma_P^{\mathsf{p}_N}, \Sigma_{P'}^{\mathsf{p}_1}, ..., \Sigma_{P'}^{\mathsf{p}_N}\right) \tag{3.2}$$

a block diagonal matrix, with $\Sigma_P^{\mathsf{p}_i}$ and $\Sigma_{P'}^{\mathsf{p}_i}$, the covariances of each of the the i-th point coordinates in each of the two point clouds, respectively.

Note that Eq. 2.16 is the result of an unconstrained minimization of the cost function $E_{dist}(\mathbf{x}, P, P')$ in Eq. 2.12, written here explicitly as a function of the transformation parameters $\mathbf{x}$ as well as the point coordinates in the two point clouds P, and P'. Then, using the implicit function theorem [33], which in general forms states that the Jacobian $\nabla\mathbf{f}$ of a function $\mathbf{f}$ satisfying a certain set of constraints $\mathbf{g}(\mathbf{x}, P, P') = 0$ can be expressed in terms of the partial derivatives of $\mathbf{g}$ with respect to $\mathbf{x}$ and (P, P') in the form

$$\nabla\mathbf{f} = -\left(\frac{\partial\mathbf{g}}{\partial\mathbf{x}}\right)^{-1}\frac{\partial\mathbf{g}}{\partial(P, P')}\,. \tag{3.3}$$

The result is used in the following way. Assume that $\mathbf{x}$ is a solution for the minimization of the cost function $E_{dist}(\mathbf{x}, P, P')$. This is equivalent to say that

$$\mathbf{g}(\mathbf{x}, P, P') = \left(\frac{\partial E_{dist}(\mathbf{x}, P, P')}{\partial\mathbf{x}}\right)^{T} \tag{3.4}$$

when evaluated at the minimizer $\mathbf{x}^*$ is equal to $\mathbf{0}^T$.

Plugging Eq. 3.4 in Eq. 3.3, the Jacobian of Eq. 2.16 becomes

$$\nabla\mathbf{f} = -\left(\frac{\partial^2 E_{dist}}{\partial \mathrm{x}^2}\right)^{-1}\frac{\partial^2 E_{dist}}{\partial(P, P')\partial \mathrm{x}} \tag{3.5}$$

Developing the summations in Eq. 2.12 and regrouping terms, the same cost function can be rewritten as a quadratic expression on the rototranslation $\mathbf{x}$

$$E_{dist} = \mathrm{x}^{\top}A\mathrm{x} - 2B^{\top}\mathrm{x} + C \tag{3.6}$$

The ensuing numerical evaluation of the partial derivatives in Eq. 3.5 becomes straight forward.

3.3 3D mapping with Pose SLAM

The Pose SLAM algorithm belongs to the variant of SLAM algorithms where only the robot trajectory is estimated and landmarks are solely used to produce relative constraints between robot poses. Pose SLAM maintains a compact state representation by limiting the number of links and nodes added to the graph using information content measures [51].

Formally, in Pose SLAM, the state vector $\mathbf{x} = [\mathrm{x}_0^\top, \mathrm{x}_1^\top, ..., \mathrm{x}_n^\top]^\top$, contains the history of robot poses from time 0 to n, which is estimated from the history of odometric observations U and proprioceptive observations Z using a canonical parameterization of Gaussian distributions

$$p(\mathbf{x}|Z,U) = \mathcal{N}^{-1}(\mathbf{x};\boldsymbol{\eta},\boldsymbol{\Lambda})\,, \tag{3.7}$$

where $\boldsymbol{\eta}$ is the information vector, and $\boldsymbol{\Lambda}$ is the information matrix.

Predictions and updates using this parametrization lead to an information filter, which compared to the traditional Kalman form, has the advantage of being exactly sparse for trajectory-based state vectors, such as ours [32].

New poses are added to the state vector as a result of the composition of odometric observations u_n with previous poses

$$\mathrm{x}_n = f(\mathrm{x}_{n-1}, u_n) = \mathrm{x}_{n-1} \oplus u_n. \tag{3.8}$$

And, for highly uneven and unpredictable terrain, such as the one in the experiments reported here, odometric data from the platform is unreliable and odometric observations are in fact computed by running the Iterative Closest Point (ICP) algorithm over two consecutively acquired point clouds.

As said, to keep the graph of poses sparse, redundant poses are not fed to the estimator. A new pose is considered redundant when it is too close to another pose already in the trajectory, and not much information is gained by linking this new pose to the map. However, if the new pose allows the establishment of an informative link, both the link and the pose are added to the map. The result is a uniform distribution of poses in the information space, as opposed to other more common methods that

trim the number of odometric relations by distributing them uniformly in Euclidean space [65].

To determine if the current pose x_n is close to any other pose in the trajectory x_i, we estimate the relative displacement between them

$$d = h(\mathrm{x}_i, \mathrm{x}_n) = \ominus \mathrm{x}_i \oplus \mathrm{x}_n \tag{3.9}$$

as a Gaussian with parameters:

$$\mu_d = h(\mu_i, \mu_n) \tag{3.10}$$

$$\boldsymbol{\Sigma}_d = \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{ii} & \boldsymbol{\Sigma}_{in} \\ \boldsymbol{\Sigma}_{ni} & \boldsymbol{\Sigma}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_n \end{bmatrix}^\top \tag{3.11}$$

where $\boldsymbol{\Sigma}_{ii}$ and $\boldsymbol{\Sigma}_{nn}$ are the marginal covariances for the state variables at stake, $\boldsymbol{\Sigma}_{in}$ is their cross correlation, and $\mathbf{H}_i$ and $\mathbf{H}_n$ are the measurement Jacobians of the relative displacement d with respect to poses x_i and x_n, respectively. Marginalizing the distribution on the displacement for each one of its dimensions we get a set of 1-D Gaussian distributions that allow to compute the probability of each variable in pose x_i of being closer than a threshold to its corresponding variable in pose x_n. If, for all dimensions, these probabilities are above a given threshold s, then pose x_i is considered close enough to the current robot pose x_n, and there is no need to include x_n in the map, unless it establishes a highly informative link.

The amount of information of a link between any two poses is decided in terms of the amount of uncertainty that is removed from the state when such link is added to the pose graph; and measured as the mutual information gain, which for Gaussian distributions is given by the logarithm of the ratio of determinants of the covariance prior to performing the state update, and after the state update is made [26, 142]. This ratio is a multiple of the number of times state uncertainty shrinks once a loop is asserted. In [51] we show, that despite being a measure of global entropy reduction, it can be computed in constant time with a single compact expression,

$$\mathcal{I} = \frac{1}{2} \ln \frac{|\mathbf{S}|}{|\boldsymbol{\Sigma}_y|}, \tag{3.12}$$

where $\mathbf{S} = \boldsymbol{\Sigma}_d + \boldsymbol{\Sigma}_y$, and $\boldsymbol{\Sigma}_y$ is the measurement covariance.

Sensor registration is an expensive process, and in practical applications, it is convenient to hypothesize whether a candidate link is informative enough before actually

aligning sensor readings. To that end, Eq. 3.12 is first evaluated using an approximation of the measurement covariance, i.e., the covariance for the individual ICP just computed in Eq. 3.1. If the result is above a given threshold, sensor registration is needed to assert data association.

When establishing such a link, the update operation only modifies the diagonal blocks i and n of the information matrix $\mathbf{\Lambda}$, and introduces new off-diagonal blocks at locations in, and ni. These links enforce graph connectivity, or loop closure in SLAM parlance, and revise the entire state, reducing overall uncertainty. The operation has linear time complexity but takes place very sparsely. Hence Pose SLAM can be executed in amortized constant time [53].

3.4 Mapping 3D scenarios

In this section we show experiments that demonstrate how the hierarchical ICP techniques for sensor registration developed in Chapter 2 and the covariance estimates for that method, developed in the previous section, can be fed as odometric as well as loop closure inputs to the Pose SLAM algorithm. In that sense, our 6DOF version of Pose SLAM [140] is applicable to build consistent dense 3D maps with range data [132]. The maps built here will be the input to the terrain classification method presented in Chapter 4, and for the path planner in Chapter 5.

We must mention, that the sensor covariance approximation computed using the first order error propagation in Sec. 3.2 was devised only for the point-to-point error metric, whilst our actual implementation of the ICP is hierarchic, using a point-to-point error metric at the coarsest levels but a point-to-plane error metric at the finest levels, weighing differently rotations than translations [129]. Nonetheless, our experiments show empirically that the computation of Σ_x using inly the point-to-point metric is accurate enough and does not jeopardize the rest of the method.

The results shown here are for mapping sessions with the same datasets as those shown in Sec. 2.3.4. Moreover, we also present results for a new dataset, captured in the interior plaza of the FME building at UPC. This new dataset encompasses a 100×40 sqm. rectangular area with various terrain types (gravel, earth, grass) and ramps. The robot used in this case is Teo, a Segway RMP 400 platform equipped with our custom

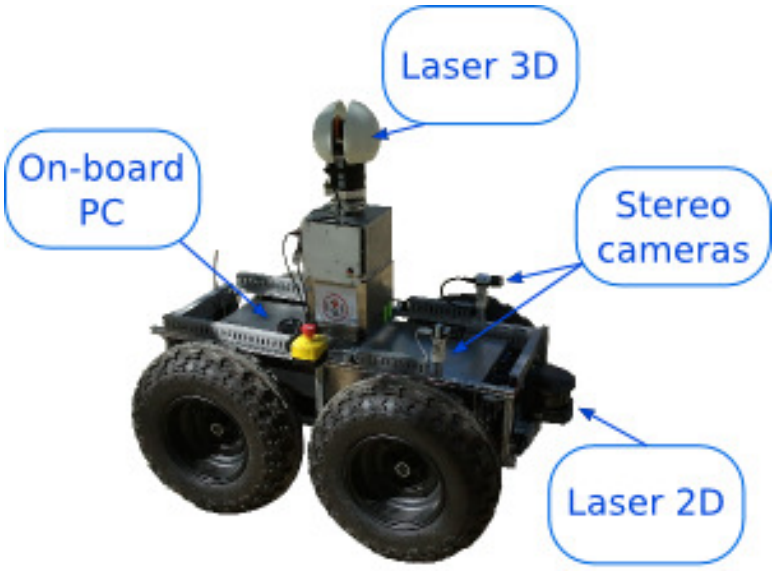
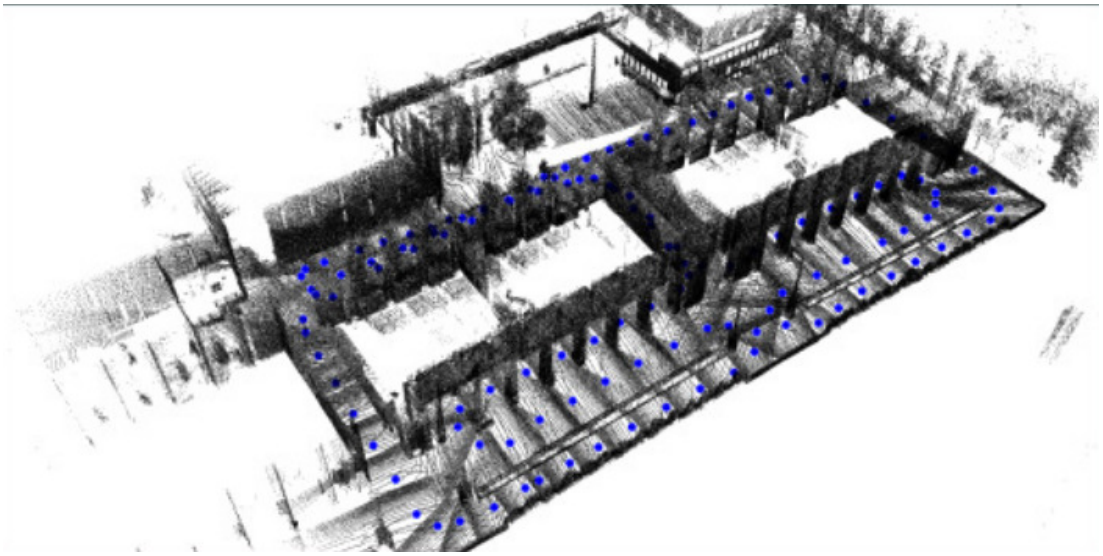
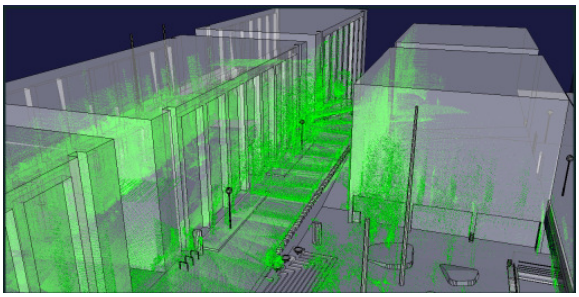


Figure 3.1: Our robot Teo.

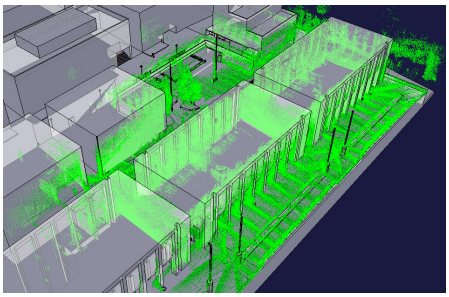
built 3D scanner, see Fig. 3.1 [132]. Each aggregated laser scan has 194,500 points with resolutions of 0.5 deg azimuth and 0.25 deg elevation and range of 30 m, with a noise level of 5 cm in depth. The Pose SLAM map built contains 30 dense point clouds with a maximum separation between consecutive poses of 18 m. In all the experiments the robot was tele-operated during data collection. The next figures show empirically the result of the SLAM method. Our intent is not to make an exhaustive numerical analysis of the mapping mechanism, but to use these maps as input to higher level navigation tasks, as shown in the following chapters.



(a)

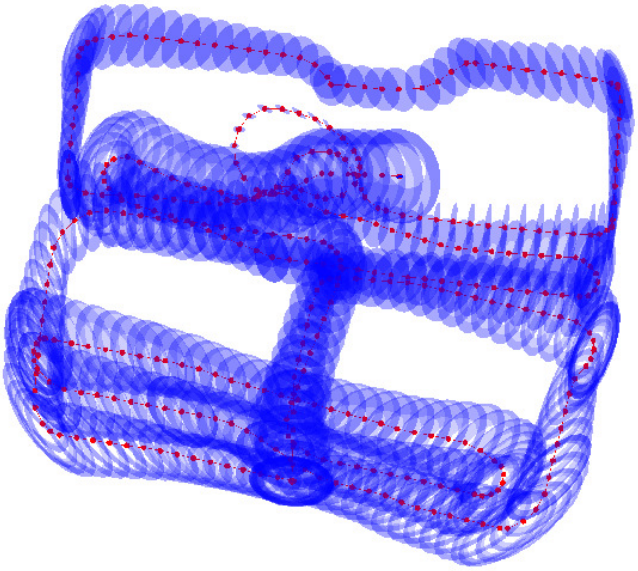


(b)

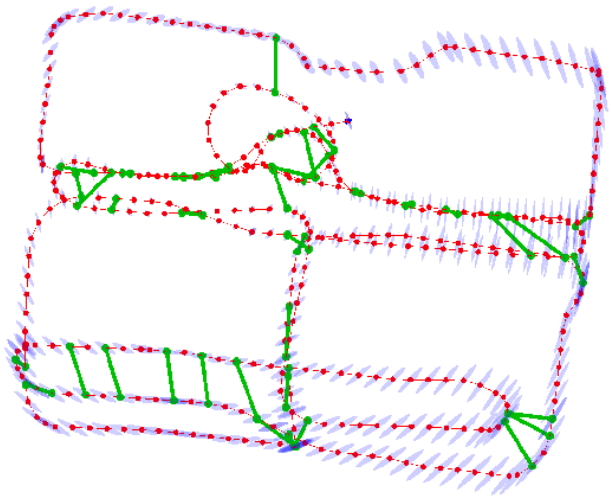


(c)

Figure 3.2: Map of the Barcelona Robot Lab computed with the hybrid hierarchic ICP approach from Chapter 2 and the Pose SLAM algorithm described in this chapter. a) Top view, robot poses are the blue points. b) and c) superposition of the final map over a virtual model of the BRL.



(a)



(b)

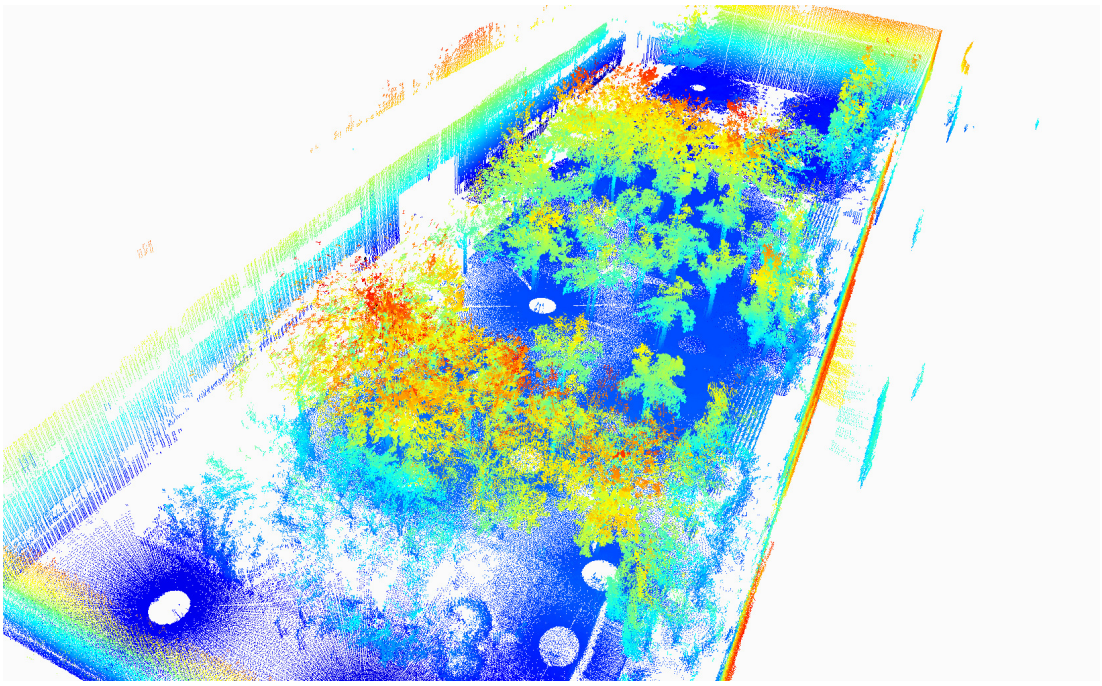
Figure 3.3: Map of the Barcelona Robot Lab, 400 poses, pose means in red and covariance hyper-ellipsoids in blue. a) Open loop point cloud registration with the proposed hybrid hierarchic ICP. b) 6DOF Pose SLAM map.

(a)

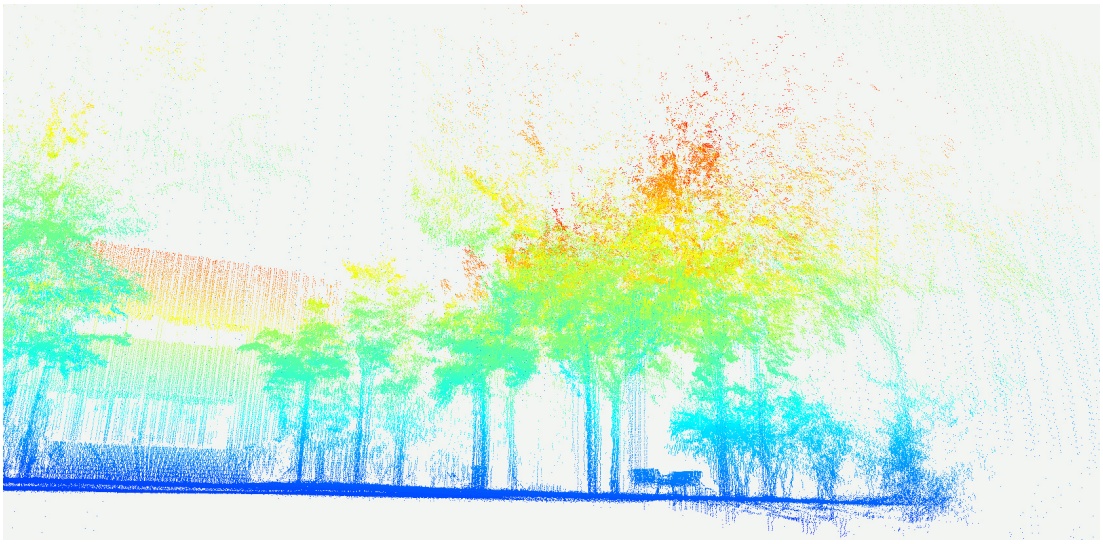
(b)

Figure 3.4: Map of the Barcelona Robot Lab a) Rendering of the initial open loop point cloud registration. b) Rendering once Pose SLAM is executed.

Figure 3.5: Example of a rich 3D point cloud rendered from the resulting Pose SLAM map of the Barcelona Robot Lab.



(a)



(b)

Figure 3.6: Computed map for the FME using the proposed hybrid hierarchic ICP for pairwise point cloud registration and Pose SLAM for global registration. (color means different height). a) Rendered top view. b) Rendered zoom in at ground level.

Chapter 4

Terrain classification

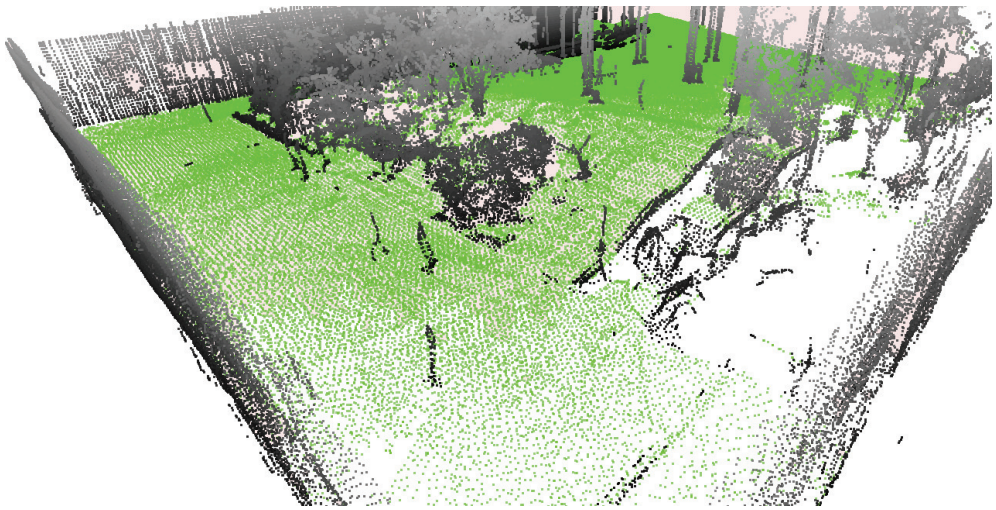
Although the capability to acquire and build dense maps is a very important task in mobile robotics [140]. If we want the robot to reliably navigate within this map, we need to identify which regions on it are safe and which are not, *i.e* the map needs to be classified.

State-of-the art terrain classification methods use 3D laser range finder data at impressively rich point cloud density, and extract features from these point clouds [28]. Such features are then used to build traversable and non-traversable classes, usually through parametric classification. Feature thresholding is a difficult task that sometimes requires human intervention [54, 125, 135]. However, learning algorithms with manual initial ground truth labeling of data can also be used [70].

In this chapter we present a pair of methods to attain high-level off-line classification of traversable areas, in which training data is acquired automatically from navigation sequences. The classification mechanisms learn traversable regions from the large 3D point maps build using the methods described in the previous two chapters. We approach the problem using Gaussian processes as a regression tool, for which the terrain parameters are learned, and also we do it for classification, using samples from traversed areas to build the traversable terrain class. These classifiers provide as output the likelihood of a point to be traversable whilst modeling few local features in a statistical manner. In a second step we filter the classes and find the traversable region edges which provide a boundary for the safe traversable regions for the robot.

To motivate the reader, Fig. 4.1 shows the type of terrain our method is able

(a)



(b)

Figure 4.1: Non-parametric classification of complex terrain. a) A picture of the Facultat de Matemàtiques i Estadística (FME) patio. b) Classification results. Traversable areas in green, and obstacles in height-valued gray. The positive samples in the training sequence used to produce this result included the points at the robot footprint that were acquired while climbing the slope next to the stairs. Consequently, the classification technique was able to correctly label such scenario.

to handle. Using only some of the traversed areas as positive training samples, our algorithms successfully classified the rest of the traversable terrain and our robot was able to climb the steep grass incline next to the stairs, despite the various types of soil in the scene (grass, dirt, gravel, etc). This is in contrast to other techniques which search for global planarity constraints or other features not in accordance with the training data. The two methods are compared against a naive parametric approach and some variants of a support vector machine. As shown in the experimental section of this chapter, our method attains f-scores –weighted average of precision and recall values– as high as 0.98 for the classification of traversable and non-traversable regions.

4.1 Related work

Terrain classification on 3D environments with non-flat surfaces has been an active research topic in mobile robotics. An early approach to the problem is the use of elevation maps [45, 46]. This is a discrete representation of the terrain that stores surface height in its cells. Neighboring cells with a difference in height above a user-defined threshold are marked as non-traversable. In [105] an extension of elevation maps is given that allows to deal with vertical and overhanging objects. These were further extended to represent multiple levels with the Muti-Level Surface Maps (MLS) [135]. And later, in [54], MLS were extended to include slope information to handle possible abysses and holes.

Terrain classification also is approached with the aid of Markov random fields in [7, 44, 90], either using cameras or range data. All these mapping schemes store the 3D environment information in regularly spaced grids. Our technique does not treat annotated 2D maps, but rather full 3D data, allowing us to handle underpasses or tree shadows with ease. Another version of terrain classification that uses 3D data is through the use of 3D occupancy maps [79]. This technique however has large memory requirements. To deal with the memory requirements of 3D occupancy maps, one can resort to the use of variable resolution grids, such as octrees [125, 146]. Instead of learning the classification, the approach in [104] segmented the point clouds through a graph-based ellipsoidal region-growing process using a minimum spanning-tree and two maximum edge-weight conditions, leading to discrete regions that represent obstacles in the environment. In [88] they classify objects according to height-length-density

parameters. The above-mentioned methods to terrain classification are parametric. They offer fast terrain segmentation, but generally it is difficult to find the rules which work for a variety of terrain types and dangerous areas like abyss.

Non-parametric approaches use some learning strategies that include the use of training data. Some recent methods use learning from demonstration [120], imitation learning [119], learning from proprioceptive measurements [6, 123], or coordinate ascent learning to estimate terrain roughness [124]. In [69] for instance, Bayesian classification is performed on elevation maps using stereo vision to compute occupancy maps. In a method similar to ours in spirit, [62] produce a grid with traversable regions from an autonomous data collection along the robot path. Their method extracts 13 different features from monocular images, and feeds them to a majority voting process to predict the traversability of each cell. Our method is simpler in that we use less features, and are not constrained to regularly spaced grid settings. Another non-grid based representation for 3D data is presented in [70]. They perform terrain classification directly over the point clouds just as we do, learning the terrain feature distribution by fitting a Gaussian Mixture Model using the Expectation Maximization algorithm on a set of hand labeled training data. We resort instead to the use of Gaussian process for classification, and do it over automatically acquired positive training sequences.

Gaussian Process (GP) [110] have recently called attention in mobile robotics for classification [91]. Compared to a more traditional technique such as Support Vector Machines (SVM) GPs offer several advantages such as learning the kernel and regularization parameters, integrated feature selection, fully probabilistic predictions and interpretability. In comparison with Support Vector Regression (SVR), GP's take into account the uncertainties, predictive variances and the learning of hyper-parameters. Using GP, [59] proposed a mobility representation where in contrast of the traversability criterion, the maximum feasible speed is used to classify the world adding exteroceptive or proprioceptive states augmenting the GP input vector. An approach using GPs on two-dimensional range data to compute occupancy probability is presented in [99]. Another approach, [92], takes advantage of GPs to build cost maps for path planning from overhead imagery. In [141], the authors infer missing data in 3D range maps for open pit mines. And recently, GP's were used to segment point clouds in not one but multiple classes [103]. In [28], the Gaussian Process Incremental Sample Consensus

(GP-INSAC) algorithm is presented, it consists of an iterative approach to probabilistic, continuous ground surface estimation, for sparse 3D data sets that are cluttered by non-ground objects. This approach however does not specifically address traversable region classification as presented in the next sections.

4.2 Gaussian processes for off-line terrain classification

We need to classify the terrain of large maps to perform navigation. The main idea behind terrain classification in mobile robotics, is to identify two terrain classes: traversable regions and obstacles. This means that we must be able to identify different environment characteristics such as: slopes, bumps, vegetation, buildings and different terrain surfaces (e.g., soil, grass, concrete), as navigable or not. For this end we will use Gaussian processes. They are a powerful mathematical non-parametric tools for approximate Bayesian inference and learning, they provide a principled, practical, probabilistic approach to learning in kernel machines [110].

Supervised learning, is the problem of learning input-output mappings with a training dataset. Self-supervised learning in robotics is refereed when there is not human intervention to build the training data set. Learning can be divided in two cases: 1) regression, concerned with the prediction of continuous quantities, and 2) classification, where the outputs are discrete class labels. A Gaussian process (GP) [110] is a Bayesian regression technique that trains a covariance function to model a non-parametric underlying distribution. Gaussian processes are non-parametric approaches in that they do not specify an explicit practical model between the input and output. The GP's are characterized by a mean function $m(\mathsf{x})$ and the covariance function $k(\mathsf{x}, \mathsf{x}')$, such that the GP is written as $f(\mathsf{x}) \sim \mathcal{GP}(m(\mathsf{x}), k(\mathsf{x}, \mathsf{x}'))$.

To solve the problem we need prior knowledge of the environment, i.e. the training dataset $\mathcal{D} = \{(\mathsf{x}_i, \mathsf{y}_i)|i = 1, ..., n\}$, where we denote x as the training input of features with dimension d, and y is a scalar output or target which is dependent of the problem to solve (regression or classification). This knowledge can either be acquired from previous data of a similar environment, or using a subset of the new data to train the GP. This algorithm employs the Occam's razor principle to avoid over-fitting, using $\mathcal{D}$

to shape a covariance function $k(\mathsf{x}, \mathsf{x}')$ into a function that characterizes the correlation between points in the dataset.

The choice of the covariance function directly impacts in the quality of predictions produced by the Gaussian Processes. The learning problem in Gaussian Processes is precisely the issue of finding the suitable free parameters for the covariance function. These free parameters are called the hyper-parameters ϕ, they are adjustable variables that give us a model of the data, such smoothness, interpreting characteristic length-scale and variance of the GP functions. The hyper-parameters are learned using the training data set $\mathcal{D}$, to later predict new inputs x_* that we may have not seen in $\mathcal{D}$.

4.2.1 Regression analysis

Regression analysis is a statistical technique for estimating the relationships among variables. It is needed to make inferences about the relationship between inputs and targets, i.e. the conditional distribution of the targets given the inputs (but there is no interest in modeling the input distribution itself). The inferences are made about the relationship between the knowns inputs x and its outputs y, in the absence of knowledge to the contrary it is assumed that the average value over the sample functions at each x is zero. Under the GP's all targets y are considered as jointly Gaussian distributed $p(\mathsf{y}_1, .., \mathsf{y}_n | \mathsf{x}_1, .., \mathsf{x}_n) \sim \mathcal{N}(\mu, \sigma)$, this is a distribution over functions and inference takes place directly in the space of functions. The mathematics below are based on [110].

When introducing the noise term, the joint distribution of the observed target values and the function values at the test locations under the prior is:

$$\begin{bmatrix} \mathsf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \tag{4.1}$$

where $X = \{\mathsf{x}_i\}_{i=1}^n$ is the matrix of training inputs (it is called the design matrix), and X_* is the test inputs matrix.

By deriving the conditional distribution corresponding to Eq. 4.1, the key predictive equations for Gaussian process regression are obtained

$$\mathbf{f}_* | X, \mathsf{y}, X_* \sim \mathcal{N}(\bar{f}_*, cov(\mathbf{f}_*)) \tag{4.2}$$

Now, for simplicity we use $K = K(X, X)$ to represent the matrix of covariances between the training points and $\mathbf{k}_* = k(X, X_*)$ to denote the vector of covariances between the test point and the training points, using this compact notation and for a single test point x_* we get:

$$\bar{f}_* = \mathbf{k}_*^T (K + \sigma_n^2 I)^{-1} \mathbf{y} \tag{4.3}$$

$$\mathbb{V}[f_*] = k(\mathsf{x}_*, \mathsf{x}_*) - \mathbf{k}_*^T [K + \sigma_n^2 I]^{-1} \mathbf{k}_* \tag{4.4}$$

where σ_n^2 is the variance of the global noise.

4.2.2 Least-squares classification

In probabilistic classification test predictions take the form of class probabilities. Unfortunately, the solution of classification problems using Gaussian processes is rather more demanding than for the regression problems, this is due that the targets are discrete class labels.

In the Gaussian processes classification (GPC) its is assigned to an input x a pattern to one of the C *classes*, $\mathcal{C}_1, ..., \mathcal{C}_C$. Classification problems can be either binary ($C = 2$ or multi-class $C > 2$. We are interested in the first case, where classification targets are labeled as $\mathsf{y} = \pm 1$. For this case one simple idea is to turn the output of a regression model into a class probability using a response function (the inverse of a link function), "squashing" a linear function of the Gaussian predictive probability through a cumulative Gaussian with the use of a (parameterized) sigmoid function as in Platt [107] using a 'Leave-One-Out' (LOO) approach,

$$p(\mathsf{y}_i | \mathbf{X}, \mathbf{y}_{-i}, \phi) = \Phi \left(\frac{\mathsf{y}_i (\alpha \mu_i + \beta)}{1 + \alpha^2 \sigma_i^2} \right) \tag{4.5}$$

where the sigmoid function has hyper-parameters (α and β), $\Phi(\cdot)$ is the cumulative unit Gaussian, $\mathbf{y}_{-i}$ is the vector of all of the training data excluding the point $(\mathsf{x}_i, \mathsf{y}_i)$, μ_i and σ_i^2 are the predictive mean and variance at the point x_i, and ϕ stands for the trained hyper-parameters of the covariance function.

The expressions for the LOO predictive mean and variance become

$$\mu_i = \mathsf{y}_i - \frac{[K^{-1}\mathbf{y}]_i}{[K^{-1}]_{ii}} \tag{4.6}$$

and

$$\sigma_i^2 = \frac{1}{[K^{-1}]_{ii}} \ . \tag{4.7}$$

Using this probability distribution, the environment can be classified into regions applying user-defined thresholds, which depend on the desired level of greediness.

In the case of classification the likelihood $p(\mathsf{y}_i|f_i)$ is not Gaussian, hence there is not closed form solution, therefore, it is needed to do approximations. In this work we use the probit likelihood for binary classification [110], this makes the posterior (given by Bayes's rule) analytically intractable. The Laplace (LP) approximation method or Expectation Propagation (EP) are models for non-Gaussian likelihood functions to classify. We select the EP over the LP approximation, since in [110] the show that LP can be overly cautious.

4.2.3 Gaussian process training

To find the best model for our data we need to train the Gaussian processes. Training is performed computing the likelihood of the hyper-parameters given the training data set $\mathcal{D}$. The optimization of the hyper-parameters ($\phi = (\sigma_f^2, \Sigma, \sigma_n^2)$) is a crucial aspect of the Gaussian process. They are very important in that they ensure that the covariance accurately captures the extent of the correlation in the environment. Under Baye's theorem hyper-parameters are computed maximizing a log-marginal likelihood function given by

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f} \tag{4.8}$$

resulting in

$$\ln p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 I)^{-1} - \frac{1}{2}\ln|K + \sigma_n^2 I| - \frac{n}{2}\ln 2\pi \ . \tag{4.9}$$

The advantage of the marginal likelihood is that it incorporates a trade-off between model fit and model complexity. A function which overfits the data leads to poor inference and large uncertainties, meanwhile, an over-generalized outcome can result in a likelihood function which chooses to ignore many of the data points in favor of adopting a less responsive behavior.

4.2.4 Covariance function choice

There are numerous covariance functions (kernels) that can be used to models the relationship between the random variables corresponding to the given data. The choice of the covariance function has a direct influence in the quality of the the predictions produced by the GP. The most common covariance function is the squared exponential, but this function tends to smooth the probabilities. Another function is the the neural network covariance

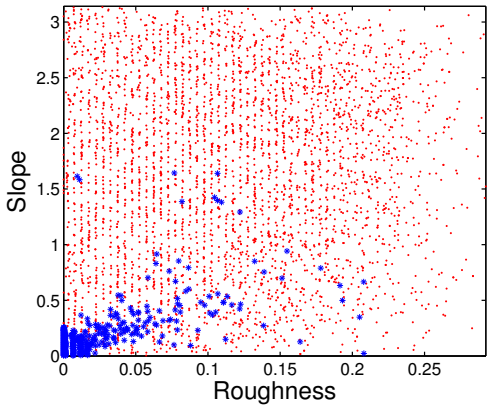
$$k(\mathsf{x}, \mathsf{x}_*) = \sigma_f^2 \arcsin\left(\frac{2\tilde{\mathsf{x}}^T \Sigma \tilde{\mathsf{x}}_*}{\sqrt{(1+2\tilde{\mathsf{x}}^T \Sigma \tilde{\mathsf{x}})(1+2\tilde{\mathsf{x}}_*^T \Sigma \tilde{\mathsf{x}_*})}}\right) \tag{4.10}$$

where $\tilde{\mathsf{x}}_i = (1, \mathsf{x}_i)^T$ is the so-called feature augmented input vector [110]. Its hyper-parameters $\Sigma = diag(\sigma_0^2, \sigma^2)$ have been set as those fitting the squared expected value of the error function on the samples x; and σ_f^2 is the hyper-parameter signal variance, used to scale the correlation between points.

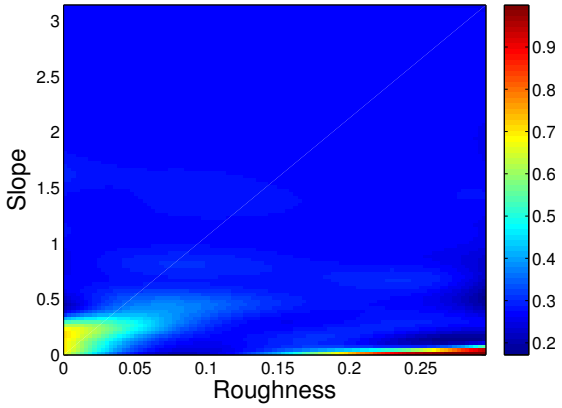
The use of a neural network covariance kernel has been proved to be effective in handling discontinuous (rapidly changing) data [141], and this is the main reason why we think it is effective to model complex terrain data. Figure 4.2 shows the likelihood for (b) the neural network covariance and (c) the squared exponential covariance. Notice how the neural network provides a smoother fit to the sampled traversable points shown in frame (a) in the figure.

4.2.5 Off-line classification

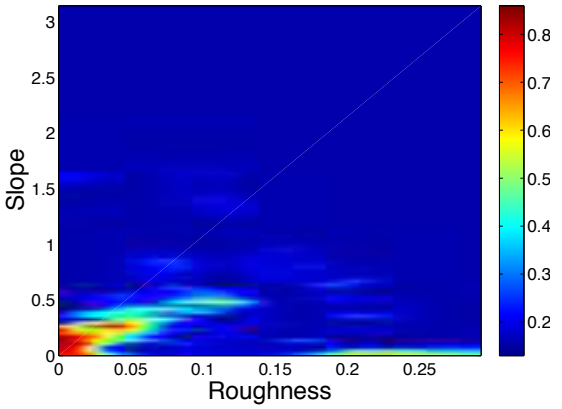
The prior knowledge of the environment is the training dataset $\mathcal{D} = \{(\mathsf{x}_i, \mathsf{y}_i) | i = 1, ..., n\}$, where x_i contains the slope and roughness input values for each point p_i in the map, and y_i is a scalar target value which is dependent of the problem to solve (regression or classification), e.g., the traversable and non traversable classification class labels. In our case, we take advantage of automatic data collection to build the training set, similar to [62]. Since our robot trajectories were human-driven, they already satisfy any unmodeled traversability constraint. The points located below the robot footprint at each node in the Pose SLAM path are thus tagged as traversable, see Fig. 4.3. This collection method alleviates the hard work of manually labeling the training dataset. For each point in this set we compute our slope and texture features, and aggregate



(a)



(b)



(c)

Figure 4.2: a) Training dataset, traversable points are in blue and obstacles in red. Likelihood plot for: b) Neural network covariance and c) Squared exponential covariance. In the color bar, the red color indicates the maximum likelihood for a point to be traversable.

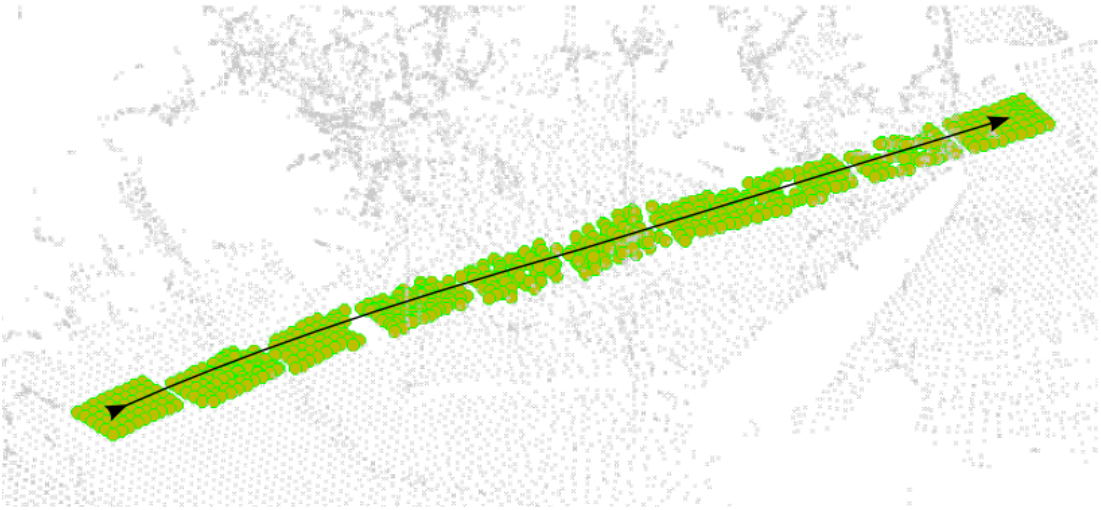


Figure 4.3: Example of the automatic data collection. In green the positive traversable samples along the robot's path footprint.

them into a positive feature vector $\mathsf{x} = [\mathsf{x}_1, \mathsf{x}_2, ..., \mathsf{x}_n] : \mathsf{x}_i = [\mathsf{f}_r, \mathsf{f}_s]$. The terrain roughness f_r corresponds to the smallest eigenvalue from the local fitted planar path computed as in 2.3.1. While the slope is the angle difference between the point's orientation (*i.e* the eigenvector associated with the smallest eigenvalue of the local planar patch) and the global reference frame. Ideally, the training dataset would consist of all the observations made. However we uniformly subsample the data. Data reduction is important since the GP has worst case computational complexity $O(n^3)$, and memory complexity $O(n^2)$.

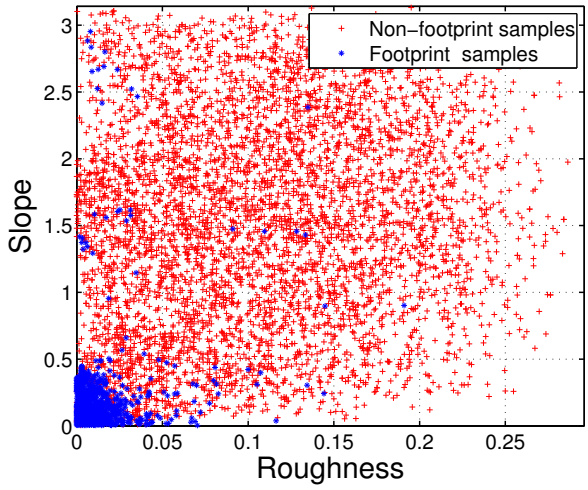
At this point we propose two different approaches to classify the points as traversable P_t or obstacles P_o. The first proposal employs GP regression and only traversable samples. The main advantage of using regression is to have less computational effort for the training and inference than GP classification, but with lower accuracy. In this case, the training dataset $\mathcal{D}$ will consist of a uniformly sampled and filtered version of x including only traversable points as in Fig. 4.3. We use a statistical density filter to remove spurious data which may affect the regression. The specification of the prior is important, because it fixes the properties of the functions considered for inference. In this case, we use a zero mean Gaussian prior [110]. The algorithm will produce a

classification threshold $y = c$ such that traversable points are $P_t = \{\mathsf{p}_i | \mathsf{y}_i \geq c\}$ and obstacle points are $P_o = \{\mathsf{p}_i | \mathsf{y}_i < c\}$. We can use regression in our proposal since data dimensionality is small and we can assume continuity, and also because we only have two classes.

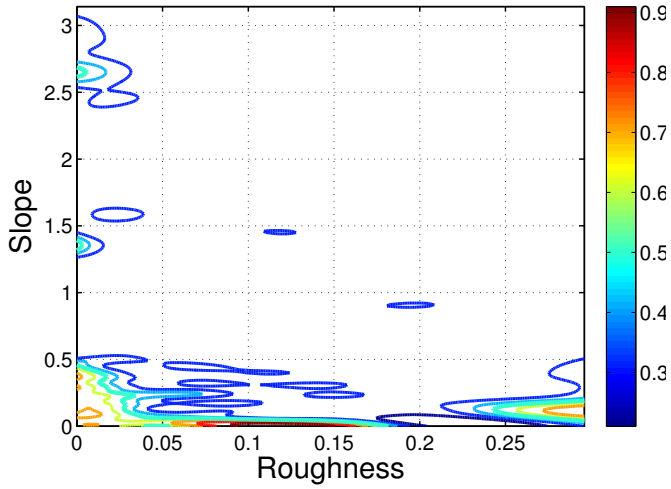
A second approach is to perform GP classification, which increases the computational load, but we believe it offers a more accurate prediction for this problem, as shown in the experiments section. The training data set is established differently than for regression. Now the training dataset needs two different classes. The first class comprises the same labeled traversable features used during regression, which came from the robot footprint samples during manual robot motion; however, in this case we do not further apply the statistical density filter in the training set because the prediction of the GP classification is expected to remove spurious data during classification. We call the input data set in this class x_+, and receives the label $\mathsf{y} = +1$. The second class with label $\mathsf{y} = -1$ is built by sampling the remaining unlabeled points x_u in x. We first sampled x_u uniformly, and then picked n random samples, where n is a multiplier related with the number of elements in x. We decided to do this because we needed to reduce the training data set dimension to ease computational burden.

Now we build the two-class training data set as $\mathcal{D} = \{(\mathsf{x}_+ \cup \mathsf{x}_-), \mathbf{y}\}$. To define the traversable points probability threshold, we create a grid over the slope and roughness features. This grid is used to compute the log predictive probability curves with the learned hyper-parameters, see Fig.4.4. It should be noted that the region of high likelihood for high roughness and low slope (on the bottom right of the Fig.4.4) is largely underrepresented in the training set, meaning that the GP would classify samples in that region with large uncertainty values. In reality, since no points in the entire dataset satisfied such condition, it is very improbable that during execution, a point would be found to have such feature values and be misclassified. Traversable points will be those with a probability larger than a threshold p_t indicated by the likelihood curve that best fits x, i.e., traversable points are $P_t = \{\mathsf{p}_i | \ln p \geq p_t\}$ and obstacle points are $P_o = \{\mathsf{p}_i | \ln p < p_t\}$.

Note that when no true negatives are available, as in our case, it is still possible to train only with positive samples during regression, or to randomly draw negative



(a)



(b)

Figure 4.4: (a) Training dataset. (b) Likelihood curves for GP classification. In the color bar, the red color indicates the maximum likelihood for a point to be traversable.

samples from the unlabeled data during classification. These two strategies are commonly used in learning, and are not unique to GPs, but are also applicable for SVM classification [31] or with the SpyEM algorithm [78].

Once the point clouds are either regressed or classified, we propose to reclassify the points and to add a filtering step to remove isolated points on each class [70]. To reclassify the points we use a median class filter on nearest neighbor information. For each point, its nearest neighbors are computed and its class revised according to the median class in the neighbor. The filter devised to remove isolated points also uses nearest neighbor information. It discards all points that have less than a constant number of neighbors in a given search radius. Finally, a statistical density filter is also applied to keep points in regions with very high density.

Identifying the edges in the traversable point set is very important to protect the robot integrity. Edge or border points may be located near abysses, on stairs and on holes that normally are out of the sensor field of view. We define edge points as $P_b = \{\mathsf{p}_i \in P_t | \mathsf{f}_{b1} \geq (\mu(\mathsf{f}_{b1}) + \kappa * \sigma(\mathsf{f}_{b1})) \wedge \mathsf{f}_{b2} \geq r \wedge \mathsf{f}_{b3} > \beta\}$, where r indicates a ratio of the PCA components, and β is the maximum angle interval for a no edge point. Next in line, we remove P_b from P_t and add these points to the obstacle set.

The first edge feature f_{b1} is given by the distance between the centroid of the search window and the query point. The second edge feature is given by $\mathsf{f}_{b2} = \lambda_1/\lambda_2$. For points located near or at a edge, $2\lambda_1 \approx \lambda_2$. To compute the third feature f_{b3}, we project the points in the query window onto the (x, y) plane, and compute the angles between the query point and all its neighbors and sort them. We use as the third feature the maximum angle interval between neighbors similar to [42]. Fig. 4.5 shows an example of edge point detection for a small portion of the FME dataset.

4.3 Point cloud classification results

In this section we show classification results for our two datasets, the BRL [131] explained in Chapter 2.4.3, and the FME dataset introduced in Chapter 3.4.

Each of the aggregated point clouds in the two maps produced with these datasets was uniformly sampled using a box size of $bz = 0.15m$. Then for each sampled point we search the nearest neighbors in a search radius of $1.35bz$ over the raw data, and

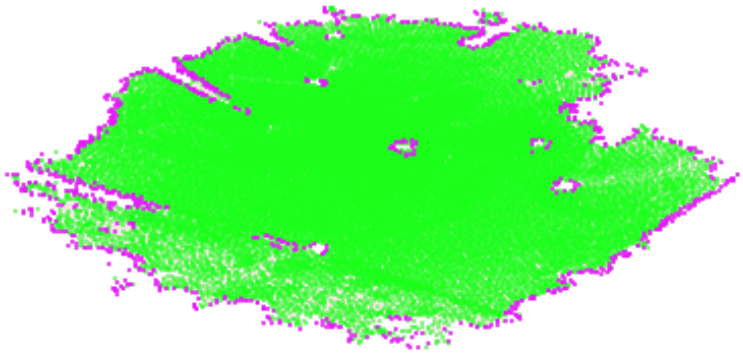


Figure 4.5: Edge classification for a small portion of the FME map.

compute f_r and f_s. Also, we discarded points with local height above $2.5m$, since the points above this height do not add useful information and are a computational burden. Filtering and border detection was also performed on the aggregated point clouds. In the outliers removal filter we discarded points with less than 7 neighbors within a $3 * h(bz)$ radius, where $h(bz)$ is the box diagonal. In the density filter, the search radius was the same. To compute border features the search radius was set to $2.5 * h(bz)$. The border detection values are: $\kappa = 2$, $r = 1.85$ and $\beta = 35^o$.

For the learning algorithm the training dataset sampling steps were 0.005 for roughness, and 0.5^o for normal orientation. In the GP classification we get the samples x_+ and x_- as stated previously. We set the number of x_- random samples as 7 times the number of elements in x_+. For the execution times reported, we run the experiments in an Intel Core i7-2720 system @ 2.20 GHZ, with 8 GB of RAM, running Ubuntu 10.04 64 bits, with MATLAB. We used the GP toolbox in [111].

We compared the two proposed GP-based segmentation schemes against a naive linear classifier, and also using SVMs with different kernel functions (linear, quadratic, polynomial, RBFs, and multilayer perception). To measure classification performance, we employ two different approaches. First we run the regression and classification methods purely on robot footprint points, and use this as ground truth to compute the recall ratios shown in Table 4.1. We can see that segmentation of the point clouds in both cases, regression and classification, produce slightly better recall values than

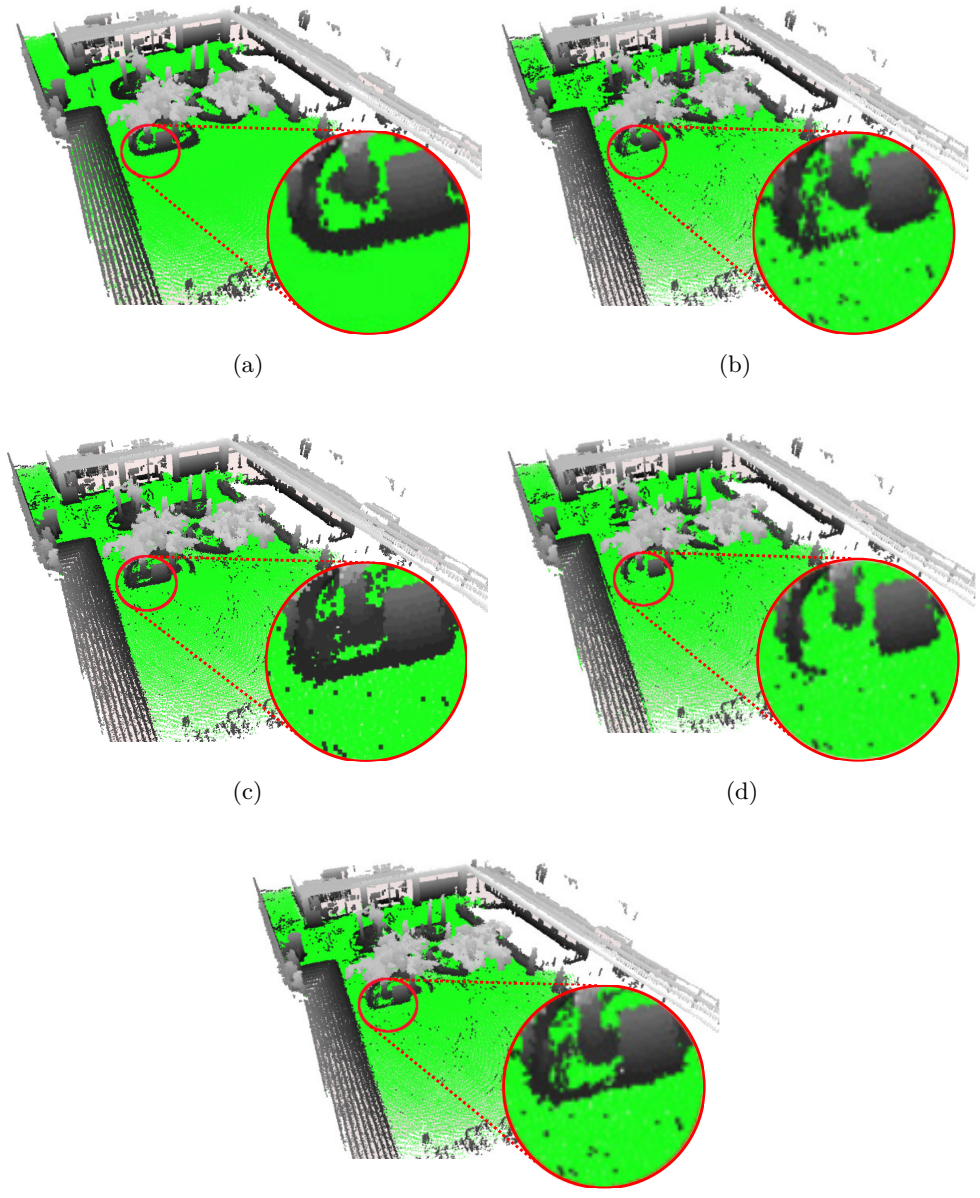
Method	FME Dataset Recall	BRL Dataset Recall
Naive parametric	0.987	0.978
SVM linear	0.9329	0.9266
SVM quadratic	0.9289	0.9266
SVM polynomial	0.9224	0.9054
SVM RBF	0.9250	0.9112
SVM MLP	0.7039	0.8417
GP regression	0.997	0.998
GP classification	0.995	0.993

Table 4.1: Traversability segmentation results using only the robot footprint to train the classifier.

naive parametric classification and significantly outperform the SVMs. By recall we mean the ratio of true positive samples over the sum of true positive and false negative samples. False negatives are in this case the points in the labeled part of the dataset that were not classified as traversable. Note that since we have only positive samples, only recall as a measure of classification performance can be computed.

The evaluation however is not completely fair. The large recall values obtained for the naive parametric classifier might be misleading since no labeled obstacles are being considered. To come up with an evaluation that takes false positive and true negative classification into account, we hand labeled the point clouds with positive and negative ground truth and computed not only recall but also precision and f-scores. The results are shown in Table 4.2. By precision we mean the ratio of true positives over the sum of true positives and false positive samples; and the f-score, a common statistic for classification performance, is computed as twice the product of precision and recall over the sum of precision and recall. The table shows how GP classification outperforms all of the other methods on f-scores in both datasets used at the expense of larger computation costs. Database sizes and computation times are given in Table 4.3.

Quantitatively speaking, GP classification is less resilient to filtering and shows better precision than regression, parametric classification, and SVM classification. In some cases however, it might be sufficient to use GP regression given its significantly smaller computational load. Fig. 4.6 shows qualitative results of all segmentation methods on the BRL dataset, compared against hand-labelled ground truth.



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

Figure 4.6: Qualitative comparison of GP traversability segmentation on the BRL dataset: a) Hand labeled classes, b) Naive parametric classification, c) SVM classification, d) GP regression, and d) GP classification. Traversable areas are shown in green and obstacles are in height-valued gray.

Method	FME Dataset			BRL Dataset		
	Precision	Recall	f-score	Precision	Recall	f-score
Naive parametric	0.9212	0.9949	0.9566	0.8832	0.9738	0.9263
SVM linear	0.8422	0.9962	0.9127	0.9015	0.9582	0.9290
SVM quadratic	0.9043	0.4761	0.6238	0.8656	0.9718	0.9266
SVM polynomial	0.8529	0.9962	0.9190	0.9213	0.3822	0.5402
SVM RBF	0.8634	0.9957	0.9248	0.8101	0.9819	0.8877
SVM MLP	0.7582	0.9886	0.8582	0.2156	0.9194	0.3492
GP regression	0.7229	0.9936	0.8369	0.5188	0.9382	0.6681
GP classification	0.9724	0.9916	0.9819	0.9112	0.9617	0.9358

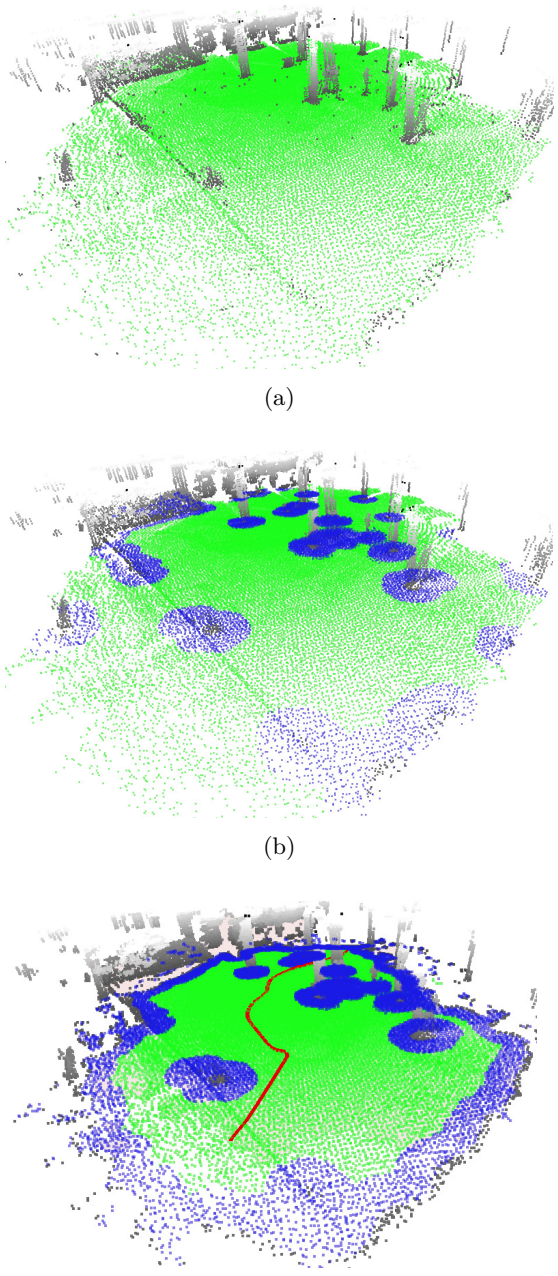
Table 4.2: Traversability segmentation results using hand-labelled ground truth.

		FME Dataset	BRL Dataset
Total size		16937 points	11518 points
GP regression	Training size	832 points	338 points
	Training time	0.8 sec	0.35 sec
	Segmentation time	0.53 sec	0.19 sec
GP classification	Training size	6080 points	4144 points
	Training time	326 sec	550 sec
	Segmentation time	11.82 sec	8.84 sec

Table 4.3: Dataset sizes and execution times.

One reason why GP classification produces slightly better results than GP regression in this particular problem might be that whereas function values take fixed values during regression, during classification, the relation from function values to class labels is achieved via class probabilities. The flexibility of using a threshold function to accommodate such class probabilities could be a reason for the improved classification results at the expense of higher computational cost. Although, the statement cannot be generalized to any other GP regression/classification problem.

Next, Fig. 4.7 shows the results from the proposed traversability classification method. Finally, Fig. 4.8 we present the classification of the complete FME and BRL.

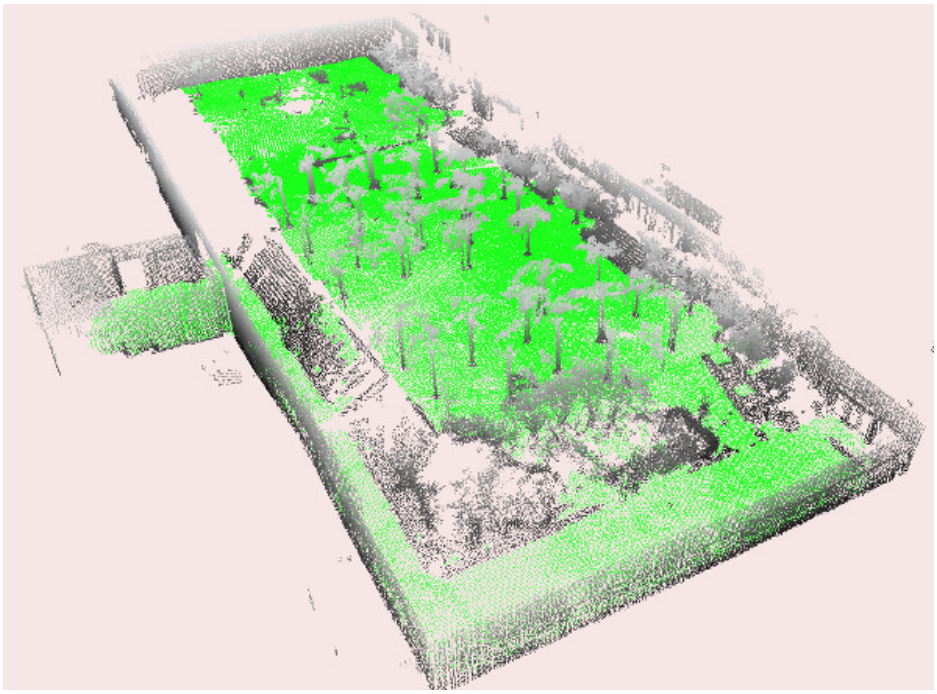


(a)

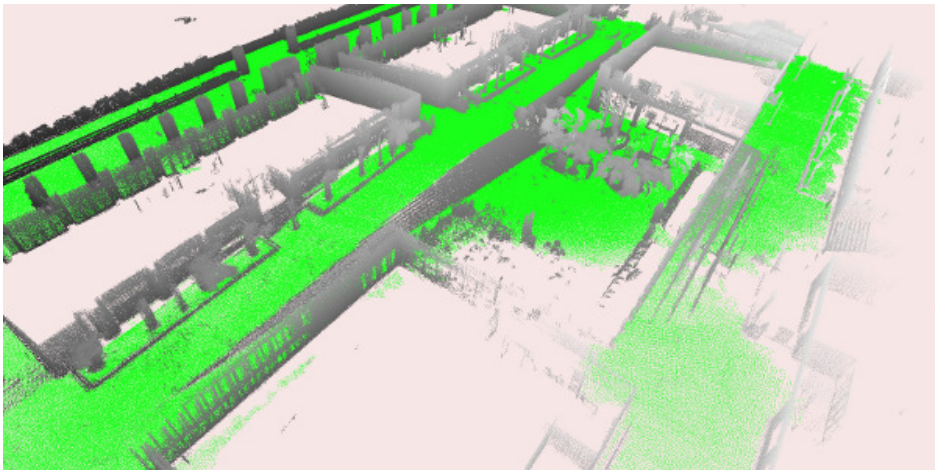
(b)

(c)

Figure 4.7: Terrain classification on the FME dataset. a) GP classifier. b) Filtered and inflated obstacles. c) Inflated obstacles and borders, and in red, a robot trajectory. Traversable points are in green, obstacles are in height-valued gray, and inflated obstacles and borders are in blue.



(a)



(b)

Figure 4.8: GP terrain classification after filtering, traversable points are in green, obstacles are in gray. a) On the FME dataset. b) On the BRL dataset.

Chapter 5

Path planning

In this Chapter we discuss how to create navigation paths for the traversable maps created in the previous Chapter. We present HRA*, a new randomized path planner that estimates safe trajectories in rich complex 3D environments. It guarantees reachability at a desired robot pose with significantly lower computation time than competing alternatives. Our path planner incrementally builds a tree using a modified version of the A* algorithm. It includes a hybrid cost policy to efficiently expand the search tree. We combine a random exploration of the space of kinematically feasible motion commands with a cost to goal metric that considers only kinematically feasible paths to the goal. The method includes also a series of heuristics to accelerate the search time. These include a cost penalty for tree nodes near obstacles, this is a penalty proportional to the inverse distance to collision; and to limit the number of explored nodes. The method book-keeps visited cells in the configuration space, and disallows node expansion at those configurations in the first full iteration of the algorithm. The performance of the method is compared against A*, RRT and RRT* in a series of challenging 3D outdoor datasets. HRA* is shown to outperform favorably against A*, RRT and RRT*, in terms of computation time, and generates significantly shorter paths than A* and RRT. For the scope of this research all the experiments models are completely known and predictable.

The rest of this Chapter is structured as follows. We start with a review to the path planning related work 5.1. Next, in section 5.2 we explain the necessary tools and propose the path planning algorithm. Finally, we present some experiments in section 5.3 .

5.1 Related work

Path planning allows robots to get from a starting point to a desired destination. The idea is to gradually compute how to move the robot into a desired goal position, so that it always avoids obstacles. Topologically, the path planning problem is related to finding the shortest path of a route between two nodes in a graph. Typically, computational complexity serves as a measure of the quality of a path planning algorithm. But apart from the computational complexity of the path planning algorithm, good motion plans are also limited by the quality/accuracy of the map and of the robot localization algorithm. The most typical path planning methods for mobile robotics include sampling-based algorithms (*i.e* RRT and PRM) and grid-based algorithms (*i.e* A* or D*) [72, 76]. Other methods for path planning include artificial potential fields [72, 112, 143], fuzzy logic [126], genetic algorithms [137], and combinatorial algorithms [76].

Grid-based search is considered by many to be the most straightforward form of path planning [72]. It is well-known that grid-based search can only obtain completeness for a fixed resolution. Grid-based path planning algorithms, such as the methods based on the A* [25], connect cells of a discretized configuration space from a start configuration to a goal configuration. To jump from cell to cell, they typically explore the action space with a deterministic minimal set of control parameters. For instance, a skid steer vehicle moving on the plane could have an action set of the form $\{-\upsilon_{max}, \upsilon_{max}\} \times \{-\omega_{max}, 0, \omega_{max}\}$, with a time interval according to the size of cells in the configuration space [10]. The time interval for which the kinematic model is unrolled can be related to the amount of clutter near the explored node, for instance, with trajectory lengths proportional to the sum of the distance to the nearest obstacle plus the distance to the nearest edge in a Voronoi diagram of obstacles [27]. Besides modifying the integration time, one can also condition the search speed according to a cost to goal heuristic. A number of heuristics can be used to modify the resolution search. This is typical of best-first algorithms such as A* in which the priority queue is sorted according to some cost to goal heuristic [49]. Another grid-based path planner is the wavefront propagation path planner [76]. An example of this planner for a 3D environment is presented in [125] for a Normal Distribution Transformation map stored in a 3D discretized space (*i.e* an octree).

Our method combines these two ideas. We propose a heuristic to modify the resolution search depending on the amount of clutter near the explored node, but instead of generating a Voronoi diagram as in [10], we add to the policy cost a penalty proportional to the inverse distance to a collision. In this way, configurations near obstacles will be sampled sparsely, whereas configurations in open space will have more chances of being tested. Furthermore, we include a measure of the distance to the goal to our cost policy, the same way as in [49], computed from Dubins paths [76], which give the shortest length path between the current configuration and the goal configuration.

Randomized sampling-based algorithms for path planning such as Probabilistic Road Maps (PRM) [60, 61] or Rapidly exploring Random Tree (RRT) [73, 75, 76] explore the action space stochastically. They randomly explore the continuous configuration space and generate a trajectory from these sampled points, they use collision detection methods to conduct discrete searches that employs these samples. Most of these approaches require a path smoothing step after the planner has generated the original path in order to get a usable path, which may not be the case of A* like algorithms. These planers guarantee a weaker notion of completeness called probabilistic completeness. This means that with enough samples, the probability to find an existing solution converges to one [76]. However, the most relevant is the rate of convergence, which is usually very difficult to establish.

A PRM planner constructs a graph in the configuration space by randomly picking configurations and attempting to connect pairs of nearby configurations with a local planner. Once the graph has been constructed, the planning problem becomes one of searching a graph for a path between two nodes [76]. By randomly exploring a continuous action space,the RRT has the property of being probabilistically complete, although not asymptotically optimal [57]. There are many variants of this algorithm [34, 67, 74, 76]. These methods can quickly explore the space but normally provide a solution far from optimal. Among them a newer version of the RRT, called RRT* [56, 58], solves this problem by triggering a rewire process each time a node is added to the tree. The rewiring process searches for the nearest neighbor in configuration space and decides whether it is less costly to move to the new node from the explored parent or from such closest neighbor. In very cluttered environments however, the RRT* algorithm may behave poorly since it spends too much time deciding whether to rewire

or not. In our method, we also sample actions randomly from a continuous action space, but in the search for neighbors within the tree, we trigger a rewiring test only when these actions reach a previously visited cell in an auxiliary maintained discretized configuration space. The RRT* also has the property of finding an optimal solution when the number of iterations tends to the infinity.

A more recent randomized motion planning strategy is cross entropy motion planning [63]. The method samples in the space of parameterized trajectories building a probability distribution over the set of feasible paths and searching for the optimal trajectory through importance sampling. Cross entropy motion planning is only applicable to problems for which multiple paths to the goal can be computed during algorithm execution.

5.2 Hybrid randomized path planning

Path planners solve the task of finding a feasible path $\tau(t) : [t_0, ..., t_n] \rightarrow \mathcal{C}_{\text{free}}$, if it exist, in a configuration space free of collisions $\mathcal{C}_{\text{free}}$, from an initial robot configuration $\tau(t_0) = \mathrm{x}_I$ to a goal robot configuration $\tau(t_n) = \mathrm{x}_G$.

Ideally, we would like to compute the minimum cost path

$$\tau^* = \arg\min(c(\tau) : \tau) \tag{5.1}$$

but since finding the optimal path can be computationally costly, we will be content with obtaining a low cost path with a reasonable computation effort. To this end, we will devise a number of heuristics aimed at pruning the search space and to bias the exploration towards the goal.

In our planning context we are faced with nonholonomic motion constraints for the vehicle, and it is through a control sequence $u(t) : [t_0, ..., t_n] \rightarrow \mathcal{U}$, $u(t_i) = (\upsilon_i, \omega_i, \Delta t_i)$, that we can move from one robot configuration to another.

5.2.1 Steering functions

We use two motion control policies. The first is the forward kinematics of the vehicle and the second one is an optimal control policy that guarantees goal reach. Our mobile

robot is modeled as a Dubins vehicle:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \upsilon \cos(\theta) \\ \upsilon \sin(\theta) \\ \omega \end{bmatrix}, \qquad |\omega| \leq \upsilon/\rho \tag{5.2}$$

where (x, y) is the vehicle position, θ is the vehicle heading, υ is the translational velocity, ω is the angular velocity, and ρ is the vehicle minimum turning radius constraint.

5.2.1.1 Robot forward kinematics

We need to estimate new robot configurations for the search tree expansion. This estimation is typically obtained by iteratively integrating the robot kinematic model for a time interval (known as dead reckoning). This estimation could be computed for instance using: the first order Euler integration (which is the most common method), the second order Runge-Kutta, or the exact method. While the Euler method is the simplest estimator, it is not very accurate when compared to others [108]. On the other hand, the exact solution is the most expensive computationally. For this reason, we selected the Runge-Kutta method. This derivation is more precise than the Euler based solution, delivering better position estimates at a lower computational cost, closer to those of the exact method, please see Fig. 5.1.

The new robot configuration after solving Eq. 5.2 using the second order Runge-Kutta method is given by:

$$\begin{aligned} x_{i+1} &= x_i + \upsilon_i \Delta t_i \cos\left(\theta_i + \frac{\omega_i \Delta t_i}{2}\right) \\ y_{i+1} &= y_i + \upsilon_i \Delta t_i \sin\left(\theta_i + \frac{\omega_i \Delta t_i}{2}\right) \\ \theta_{i+1} &= \theta_i + \omega_i \Delta t_i \end{aligned} \tag{5.3}$$

The robot motion commands (υ_i, ω_i) are subject to minimum and maximum speed constraints $\upsilon_i \in [\upsilon_{\min}, \upsilon_{\max}]$ and $\omega \in [-\omega_{\max}, \omega_{\max}] | (\omega = 0 \Rightarrow \upsilon_{\max}), (|\omega| = \omega_{\max} \Rightarrow \upsilon_{\min})$. This definition imposes the maximum speed constraint $\upsilon^2 + \omega^2 \leq 1$ [76].

We want to compute random velocity commands subject to this constraint. To avoid significant rotation at high speeds we introduce a parameter $l_u \geq 1$ designed to control the relation between the translational and rotational velocities. The mechanism to sample motion commands is as follows. We draw random translational velocities

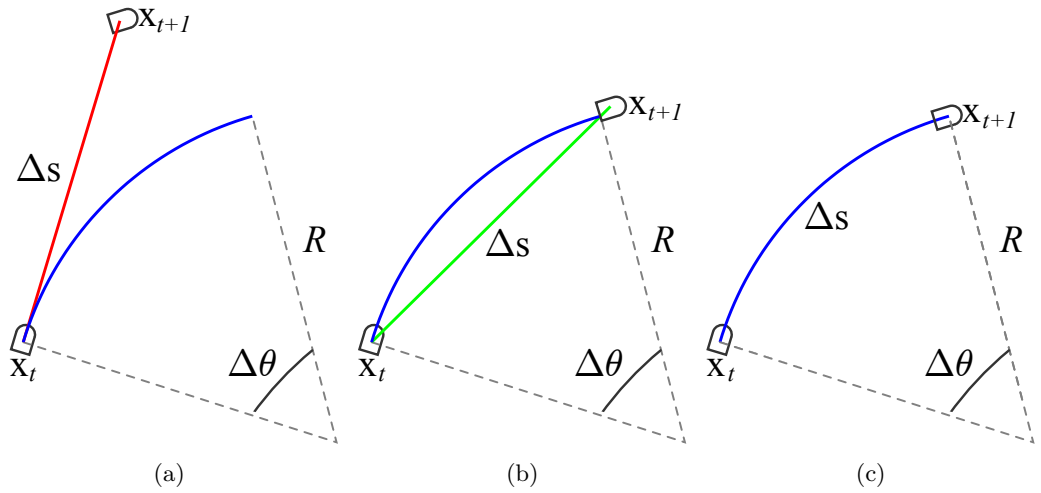


Figure 5.1: Robot forward motion model from x_t to x_{t+1} with three different methods. a) Euler (red) b) Runge-Kutta (green) and c) Exact solution (blue). Where Δs is the path length increment, $\Delta\theta$ is the orientation increment, and R is the turning radius

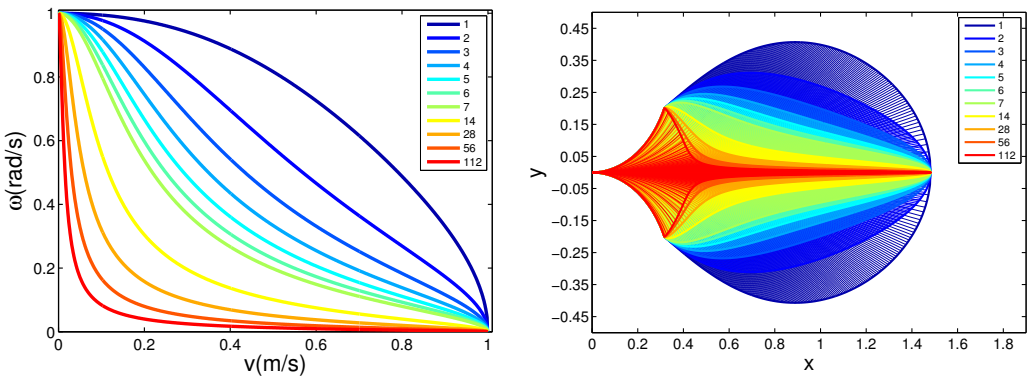


Figure 5.2: Reachable curvature sets. a) Normalized turning radius and velocity correlation for different values of l_u. b) Integrated trajectories for different values of l_u and $\Delta t = 1$)

from the fixed interval $v_{\text{rand}} = [0, 1]$ and compute its corresponding rotational velocity $\omega(v_{\text{rand}}) = \sin(\arccos(v_{\text{rand}}))$. We then scale them with:

$$\begin{aligned} v_{l_u} &= l_u \, v_{\text{rand}} \, , & y &= 1 - v_{\text{rand}} \\ r &= \sqrt{v_{l_u}^2 + y^2} \, , & \omega_{l_u} &= \sin(\arccos(v_{l_u}/r)) \end{aligned} \tag{5.4}$$

In this way, we have a gauge to modify the robot behavior. See Fig.5.2. The original velocity commands on the sphere occur at $l_u = 1$. Finally, we scale these values to the range given by the translational and rotational velocity:

$$v_i = v_{\min} + v_{l_u}(v_{\max} - v_{\min}) \tag{5.5}$$

$$\omega_i = \alpha \omega_{\max} \omega_{l_u} \, , \quad \alpha \sim \text{rand}(-1, 1) \tag{5.6}$$

Finally, the sampling period Δt_i is randomly selected from a small interval $[t_{\min}, t_{\max}]$, and the new robot pose x_{i+1} and trajectory $\tau(\Delta t_i)$ are computed by integrating Eq. 5.3 over small time steps $\delta t << \Delta t_i$.

The strategy essentially limits large curvature values at high translational speeds, producing smoother trajectories.

5.2.1.2 Locally optimal control policy

Now that we have a mechanism to sample configurations from a search node with a linked configuration x_i, we still need a way to measure the cost to reach goal completion. To that end, we compute the length of a Dubins curve from the new computed configuration x_{i+1} all the way to the goal x_G.

Dubins curves [76] are optimal paths made up of circular arcs connected by tangential line segments. With geometric arguments in his pioneer work, Dubins [29] solved the shortest path problem between two points with specific orientation in an obstacle-free plane, with two constrains: 1) the path should be tangent to the given vectors at start and goal, and 2) a bounded maximum curvature $\kappa = 1/\rho$ over the path interval. The optimal path to this problem consists of a smooth concatenation of no more than three pieces. Each of these pieces describing either a straight line, denoted by L, or a circle, denoted by C (typically C^+ describes a traveled clockwise circle, and C^- refers to a traveled counter-clockwise circle). These segments are either of type CCC

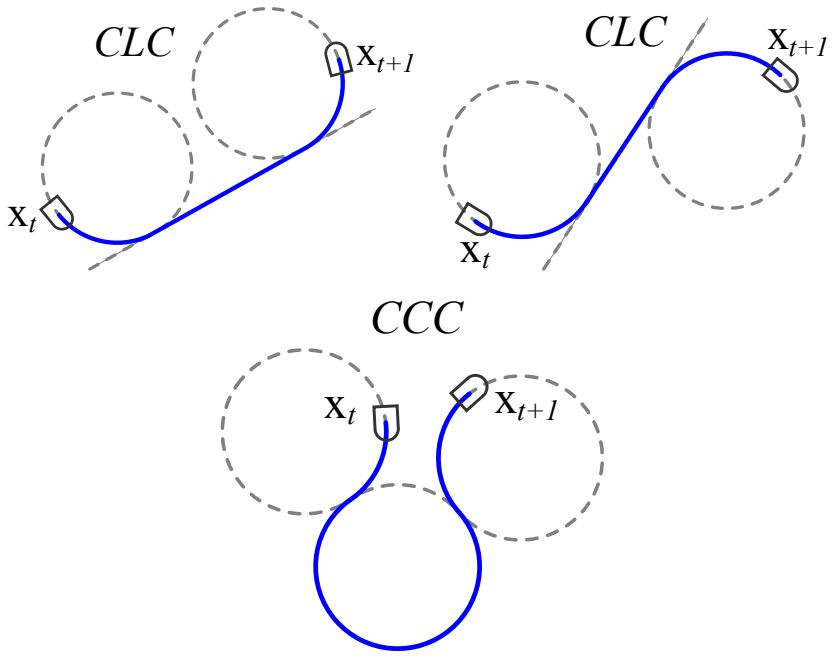


Figure 5.3: Example of Dubins path sets.

or CLC, that is, they are among six types of paths. In our application we restrict to only use the CLC type and they are specified by a combination of left, straight, or right steering inputs, leading to only four types of paths (C^+LC^-, C^-LC^+, C^+LC^+, C^-LC^-). These concatenation sequences are known as R-geodesics, and there are further constraints on how these segments may be connected together. R-geodesics are of particular interest in robotics because they describe optimal paths for a simple model of a car with bounded steering angle and velocity [76]. This paths will give us a locally optimal control policy and a locally optimal path $\tau(t)$.

5.2.2 Cost estimation

Our cost unit is time, and as in [36] we split the cost into accumulated cost (c_{acc}), *i.e* the time we have integrated the kinematic model so far to go from x_I to x_{i+1}, and a cost to the goal c_{goal}, the optimistic bound given by the Dubins curve from x_{i+1} to x_G, assuming the remaining path is free of obstacles. Similar to [68], we also use upper and lower bounds, but in a slightly different manner. The upper bound c^+ is used during node selection to improve the exploration, by adding heuristics that bound the node

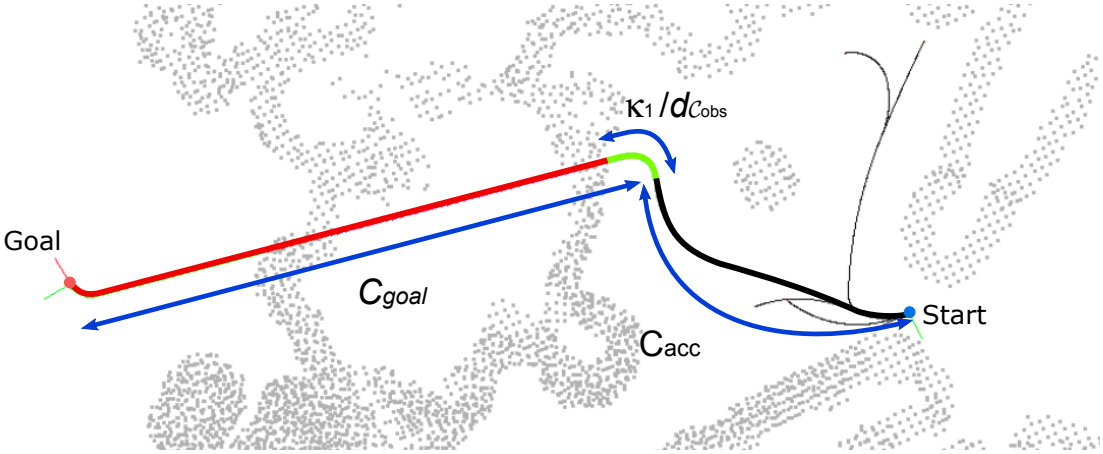


Figure 5.4: Proposed cost estimation. Bold black current branch to compute c_{acc}, red plus green lines c_{goal}, and green line h_1.

search. The lower bound c^- is used to determine the best path from the solution set, i.e, c^- will be the cost to minimize in Eq. 5.1.

The accumulated cost is computed with

$$c_{\text{acc}} = \sum_{k=1}^{i+1} \Delta t_k \ , \tag{5.7}$$

and the upper and lower bounds are computed with:

$$c^- = c_{\text{acc}} + d_{\text{goal}}/v_{\max} \tag{5.8}$$

$$c^+ = c_{\text{acc}} + d_{\text{goal}}/v_{\max} + \sum_{j=1}^{n} h_j \tag{5.9}$$

where d_{goal} is the optimal path length of the Dubins curve to the goal, and h_j are each of the heuristics we use to penalize a path.

We can use as many as desired penalization heuristics h_j to make some configurations less attractive to the planner. Our first heuristic is a local penalty inversely proportional to the distance to the nearest obstacle in front of the robot. To compute it, we ray-trace the environment at the robot's heading and measure the distance to the nearest obstacle in front of the robot $d_{\mathcal{C}_{\text{obs}}}$. We define the penalty $h_1 = \kappa_1/d_{\mathcal{C}_{\text{obs}}}$, where κ_1 is a user selected factor to weight the contribution of this heuristic in c^+.

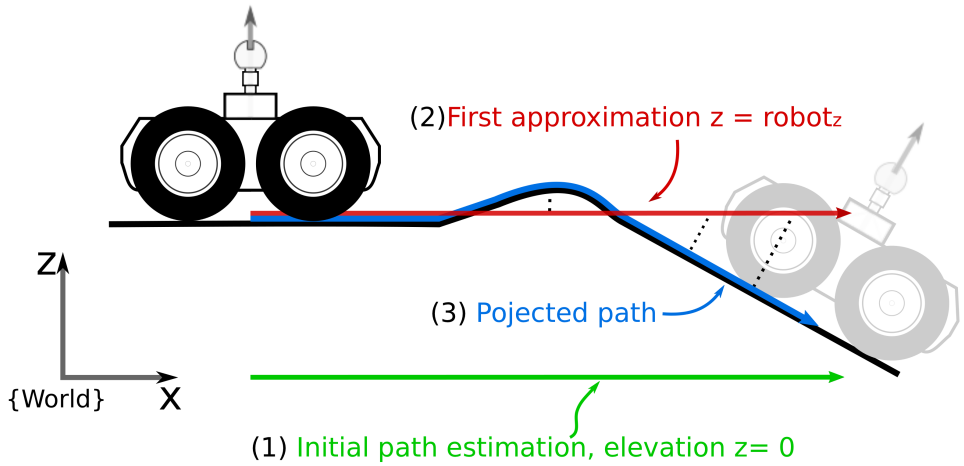


Figure 5.5: Proposed path projection step, used to estimate the local path and the final robot orientation.

Another penalty h_2 occurs when $\tau_{\text{goal}} \subset \mathcal{C}_{\text{free}}$, i.e. the Dubins path has a clear path to the goal. Then, $h_2 = \infty$ and $h_2 = 0$ otherwise. This way we avoid to explore that node again. This is a termination condition because τ_{goal} is locally optimal so it is not possible to find a better solution from that configuration to the goal.

5.2.3 Path projection

The presented control policies compute segment paths parallel to a plane. These paths could be either on a local planar patch, or with respect to a global coordinate frame. These paths however, contain no information about their elevation or orientation with respect to the 3D surface. In [39] the authors use Lagrangian interpolation to project 2D roadmap streamlines to the 3D terrain model, and a feasibility test for each 3D candidate path is computed. In [64], the robot is assumed to move at small distances to little planar patches, and robot transitions between patches are assumed with constant velocity. To compute the 3D path, we propose also a projection of the 2D local paths to the 3D surface, but in contrast to [39], we not only validate feasibility, but an iterative nearest neighbor search is used to refine the local path.

Local paths are computed parallel to the world xy plane. The elevation value for

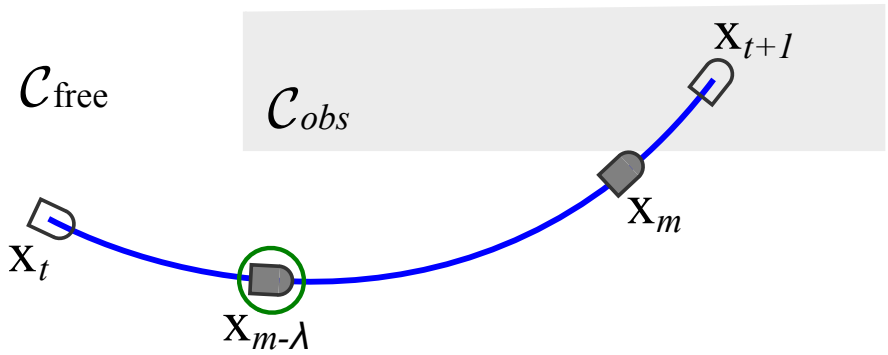


Figure 5.6: Configuration recovery from a path segment $\tau(t)$ (blue line) in collision. The last collision free configuration $\mathrm{x}_m \in \tau(t)$ is computed, then we return in time to $t_{m-\lambda}$ adding a new node with the configuration $\mathrm{x}_{m-\lambda}$.

each new configuration in the path is initialized with that of its parent node. Then, a NN search is performed to find its closest traversable point in the 3D map, substituting the z coordinate with that of the NN. If the difference between any two neighbor configurations in the 3D local path is larger than a threshold (i.e., if the hill is deemed too steep), we enter in a loop searching for a new NN assignment, see Fig. 5.5.

Finally, as in [121], we use local planar information to compute the robot orientation for the last configuration in the path. Employing the right hand rule and the local normal vector that indicates the robot orientation w.r.t the world $\hat{\mathbf{n}} = [n_x, ny, nz]^T$.

5.2.4 Collision detection

Once the local 3D path is computed, we need to check whether it remains collision free or not. If a path segment $\tau(t)$ is found to be in collision, we keep the path segment that lies in $\mathcal{C}_{\text{free}}$, i.e., $\tau(t) = [t_0, ..., t_m] \subset \mathcal{C}_{\text{free}}$, and the configuration $\mathrm{x}_{m-\lambda}$ is added to the search tree, then we update the path and the commands. At that point, a new exploration step is triggered. The parameter t_λ is introduced to avoid configurations blocked by obstacles. Notice that if $(t_m - t_{m-\lambda} - \Delta t_m) \leq 0$, the algorithm would return to collision. Keeping $\mathrm{x}_{m-\lambda}$ helps to add new nodes saving time since we benefit from recovering a part of an already computed path.

5.2.5 Tree rewiring

The rewiring procedure proposed in this work is different from that in [56, 58]. Instead of using vicinity information around a volume or the k-nearest neighbors for each new added node, we use a similar approach to that in [27]. We maintain an auxiliary grid of the configuration space $\mathcal{C}_\tau$, and mark visited cells on it. If a cell was previously visited, we compare the cost of the new and the old paths. In contrast to [27], we do allow rewiring even if the valid configuration at that cell has children. For this purpose we compute the Dubins path from the best node q_{best} to the old position in the rewire candidate node. Alg. 4 describes the rewiring process, whereas Alg. 5 overviews HRA*, our hybrid randomized path planning method.

5.3 Planning with HRA* in 3D environments

We show the performance of HRA* in three different outdoor datasets of uneven challenging terrain. The data sets where classified using the classification method proposed in the Cha. 4. The first dataset is the one acquired in the Facultat de Matemàtiques i Estadística (FME), and described in Sec. 3.4. The dataset includes 39 dense 3D scans collected on different traversed surfaces (soil, grass and gravel), our robot Teo was driven over bumps of about 10-15 cm height. The second dataset corresponds to a set of point clouds from the BRL data set [131], explained in 2.4.3. The third dataset is part of the Canadian Planetary Emulation Terrain 3D Mapping Dataset [134]. It includes five subsets that emulate planetary terrain. We use the boxmet (BM) subset which has 112 scans and covers 7200 m^2. To validate and compare our method we extracted six different scenarios from these datasets: four from the FME, and one from each of the BRL and BM datasets, please see Fig. 5.8. With all these datasets, we analyzed different situations in real static environments such as: narrow passages, cluttered obstacles, bay-like situations, ramps, forks and more.

For the RRT and RRT* algorithms, we ran 50 Monte Carlo simulations with a maximum number of iterations $i = 20,000$ for each of the five scenarios.

For the A* algorithm, we used 7 motion primitives as described in Sec. 5.2.1. In this case, experiments were run only once since the A* solutions are unique. We set

TreeRewire($\mathcal{T}, q_{best}, u_k, \mathbf{x}_k, q_j, \mathbf{X}$)
Input:
$\mathcal{T}$: Tree.
q_{best}: Current best node, node in expansion.
$\mathbf{u}_k$: Estimated control commands.
$\mathbf{x}_k$: Estimated robot position.
q_j: Node at position $\mathbf{x}_k$.
$\mathbf{X}$: 3D map.
Output:
$\mathcal{T}$: Updated Tree.

1: **if** $\sim \mathcal{T}(q_j).blocked$ **then**
2: **if** $(q_j \neq q_{best}) \wedge (\mathcal{T}(q_j).parent \neq q_{best})$ **then**
3: $\mathcal{T}(q_j).c_{acc} \leftarrow$ GetCost($\mathcal{T}(q_j)$)
4: $\mathcal{T}(q_j).rewire = 2$
5: **if** HasChildren(q_j) **then**
6: $(\tau_k, c_{new}) \leftarrow$ GetDubins($\mathcal{T}(q_{best})$.x, $\mathcal{T}(q_j)$.x)
7: **if** CollisonCheck $(\tau_k, \mathbf{X})$ **then**
8: **return** $\mathcal{T}$
9: **end if**
10: **else**
11: $\mathcal{T}(q_j).rewire = 1$
12: $c_{new} = \mathbf{u}_k.t$
13: **end if**
14: $c_{accnew} = c_{new} + q_{best}.c_{acc}$
15: **if** $c_{accnew} > \mathcal{T}(q_j).c_{acc}$ **then**
16: **if** HasChildren($\mathcal{T}(q_j)$) **then**
17: $\mathcal{T} \leftarrow$ UpdateNode($\mathcal{T}, q_j, q_{best}, c_{new}$)
18: $\mathcal{T} \leftarrow$ UpdateChildren($\mathcal{T}, q_j, c_{new}$)
19: **else**
20: $\mathcal{T} \leftarrow$ UpdateNode($\mathcal{T}, q_j, q_{best}, c_{new}, \mathbf{x}_k$)
21: **end if**
22: **return** $\mathcal{T}$
23: **end if**
24: **else**
25: **return** $\mathcal{T}$
26: **end if**
27: **else**
28: **return** $\mathcal{T}$
29: **end if**

Algorithm 4: Tree rewire

HybridRandomizedPathPlanner($\mathrm{x}_I, \mathrm{x}_G, I, K, \mathbf{X}$)
  Input:
    x_I: Initial configuration.
    x_G: Goal configuration.
    I: Maximum number of iterations.
    K: Number of motion samples per iteration.
    $\mathbf{X}$: 3D map.
  Output:
    τ^*: Path.
    $\mathbf{u}^*$: Commands.

1: $\mathcal{T}.init(\mathrm{x}_I, \mathbf{X})$
2: $\mathcal{C}_{\text{free}} \leftarrow$ ComputeDiscretizedCspace($\mathbf{X}$)
3: $\mathcal{C}_\tau \leftarrow$ UpdateAuxiliarCspace($\emptyset, \tau_0$)
4: $solution = 0$
5: **for** $i = 1$ to I **do**
6: $q_{best} \leftarrow$ GetBestNode($\mathcal{T}$)
7: $u \leftarrow$ GenerateRandomCommands(K)
8: **for** $k = 1$ to K **do**
9: $(\mathbf{x}_k, \tau_k) \leftarrow$ ForwardKinematics($\mathcal{T}, u_k, q_{best}$)
10: $m \leftarrow$ CollisonCheck($\tau_k, \mathcal{C}_{\text{free}}, \mathbf{X}$)
11: **if** m **then**
12: $(\mathbf{x}_k, \tau_k, u_k) \leftarrow$ UpdateData($m, \mathbf{x}_k, \tau_k, u_k$)
13: **end if**
14: **if** $\sim solution$ **then**
15: $visited \leftarrow$ AreVisitedCells(τ_k)
16: **end if**
17: **if** $(\sim m) \wedge (\sim visited)$ **then**
18: $q_j \leftarrow$ GetNodeInCell($\mathbf{x}_k, \mathcal{C}_{\text{free}}$)
19: **if** q_j **then**
20: $\mathcal{T} \leftarrow$ TreeRewire($\mathcal{T}, q_{best}, \mathbf{x}_k, q_j$)
21: **else**
22: $\mathcal{C}_\tau \leftarrow$ UpdateAuxiliarCspace($\mathcal{C}_\tau, \tau_k$)
23: $c_{\text{acc}} \leftarrow$ IncrementCost($\mathbf{x}_k, \mathcal{T}$)
24: $(c_{\text{goal}}, \tau_{\text{goal}}) \leftarrow$ DubinsToGoal($\mathbf{x}_k, \mathrm{x}_G$)
25: $m \leftarrow$ CollisonCheck($\tau_{\text{goal}}, \mathbf{X}$)
26: $\mathcal{H} \leftarrow$ ComputePenalties
27: $(c^+, c^-) \leftarrow$ ComputeBounds($c_{\text{acc}}, c_{\text{goal}}, \mathcal{H}$)
28: $\mathcal{T} \leftarrow$ TreeUpdate($\mathcal{T}, \mathbf{x}_k, \tau_k, u_k, c^+, c^-$)
29: **end if**
30: **if** $\sim m$ **then**
31: $(\mathbf{u}^*, \tau^*) \leftarrow$ SelectShortestPath($\mathcal{T}$)
32: $solution = 1$
33: **end if**
34: **end if**
35: **end for**
36: **end for**
37: Return $(\mathbf{u}^*, \tau^*)$

Algorithm 5: HRA*: Hybrid Randomized Path Planning

$i = 10,000$ for the FME scenarios and $i = 20,000$ for the BRL and BM scenarios. To allow for a more fare comparison, rewiring is allowed as described in Sec. 5.2.5.

For HRA*, we run two sets of 50 Monte Carlo simulations each, $i = 10,000$ iterations, and we enable the heuristics h_1, h_2, and rewiring. In the first set, HRA*1, cell bookkeeping is disabled, whereas in the second set, HRA*2, bookkeeping is enabled. The parameter l_u limiting the ratio between translational and angular velocities was selected empirically, by testing HRA*2 for different values of l_u. A value of $l_u = 4$ is a good compromise between path quality improvement and execution time. The other parameters were chosen as $\kappa_1 = 0.1m$ and $d_{\text{goal}} = 0.3m$.

When using cell bookkeeping we consider that a path is new if the ratio between new cells and visited cells is larger than 1%. Incredibly, such a small percentage of new cells was enough to obtain a significant boost in computation speed. The discretization of the configuration space was set to $(\Delta x, \Delta y, \Delta\theta) = (0.3m, 0.3m, 5deg)$.

Experiments were run in MATLAB. Some functions were implemented in C++ and embedded as mex files. We use the ANN library [89] for the nearest neighbor search. All experiments are executed on an Intel Core i7-2720 system @ 2.20 GHZ, with 8 GB of RAM memory running Ubuntu 12.04 64 bits.

We also show first, on a synthetic scene, how the different planners perform space exploration, see Fig 5.7. We plot the state of the planner tree at 100, 250 and 500 iterations for each method before reaching the solution. Note however that HRA*2 does not take more than 250 iterations to reach the solution.

Table 5.1 shows the first and best computed path lengths for all methods, indicating the mean value of the first length obtained for each of the Monte Carlo runs, and the minimum path length obtained over all Monte Carlo runs, respectively. The table shows that the two versions of the proposed approach HRA*1, and HRA*2 are able to compute path lengths comparable to A* with rewiring in all datasets, and significantly shorter than RRT.

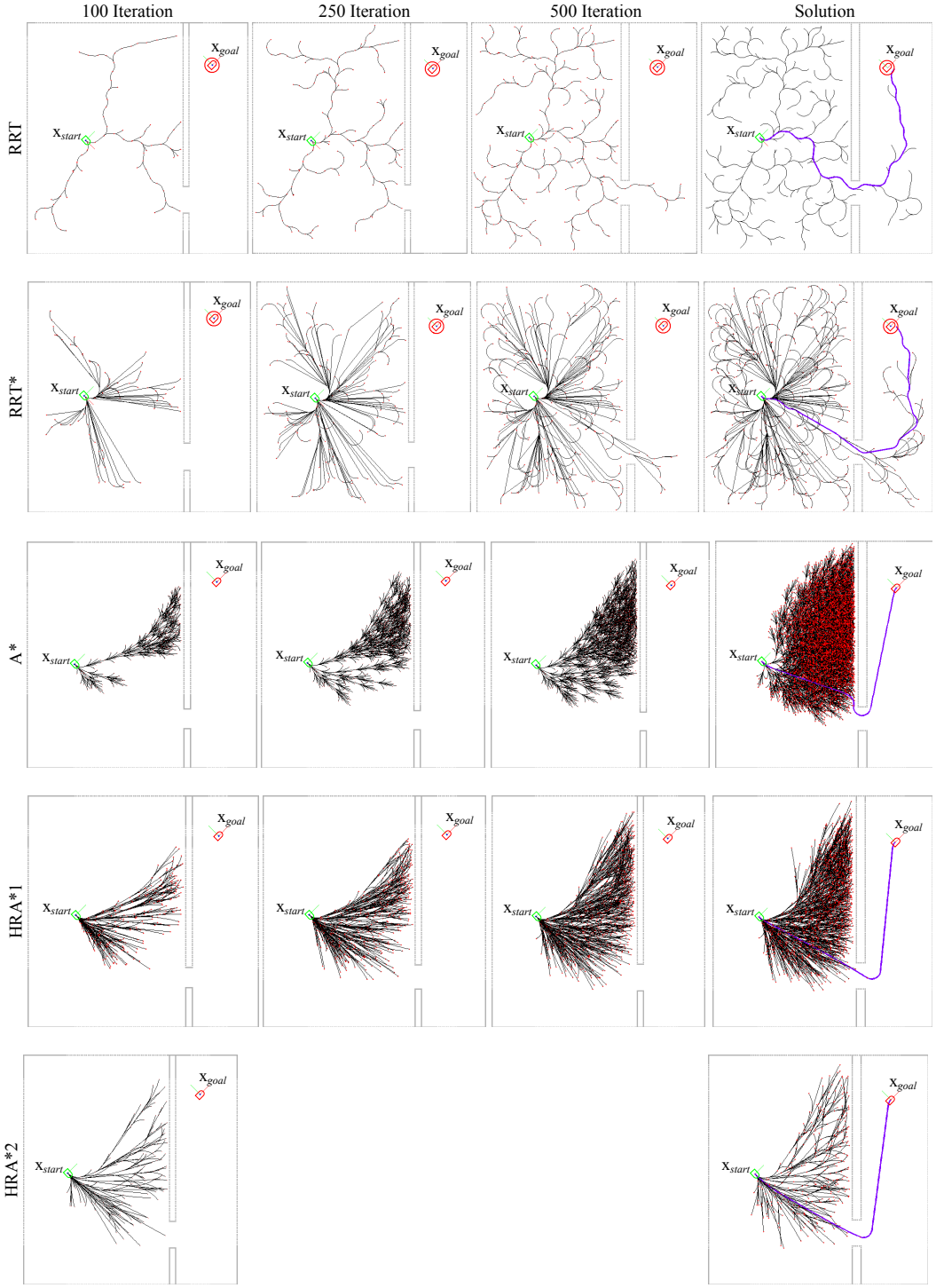


Figure 5.7: Examples of different path planners in a synthetic environment. The green robot indicates the start position and the red one the goal. Moreover, in the RRT planners we draw the goal region.

	Scenario											
	FME1		FME2		FME3		FME4		BRL		BM	
Method	First	Best	First	Best	First	Best	First	Best	First	Best	First	Best
A* with rewiring	18.12	17.63	33.84	33.93	18.61	17.84	36.97	36.47	14.77	14.77	**30.97**	30.83
HRA*1	16.52	16.34	34.54	32.92	18.29	17.39	36.48	35.98	15.64	14.46	31.03	30.50
HRA*2	16.40	15.73	35.42	32.28	18.50	17.28	36.59	35.50	15.79	14.50	31.25	30.59
RRT*	**15.58**	**14.73**	**33.48**	**30.55**	**18.10**	**16.57**	**36.35**	**33.55**	**14.76**	**13.73**	31.58	**29.60**
RRT	19.57	15.89	46.17	35.66	23.29	17.38	44.09	37.46	21.18	15.92	41.16	32.25

Table 5.1: Final path length in meters computed for all methods. First and best solutions.

	Scenario											
	FME1		FME2		FME3		FME4		BRL		BM	
Method	First	Best	First	Best	First	Best	First	Best	First	Best	First	Best
A* w/ rewiring	2.01	95.75	10.04	**10.05**	51.77	83.64	20.62	**21.93**	184.90	184.90	425.82	799.41
HRA*1	2.37	132.92	16.88	62.10	18.31	47.59	14.01	76.65	24.89	36.84	308.43	**371.68**
HRA*2	**1.20**	342.02	13.55	396.63	**6.05**	169.37	**10.51**	139.66	**4.70**	191.02	**47.14**	724.43
RRT*	113.18	2555.42	387.82	4164.13	239.80	2074.17	655.41	3538.28	271.45	1516.01	1311.05	3840.55
RRT	1.89	**25.06**	**8.58**	33.20	20.85	**41.89**	14.71	40.82	10.42	**21.54**	415.37	611.86

Table 5.2: Computation times in seconds for all methods. First and best solutions.

Note how cell bookkeeping (HRA*2) improves the best solution on almost all scenarios when compared to (HRA*1) at the expense of a possible larger first solution. The reason is that cell bookkeeping enforces sparsity in exploration gaining speed in finding the first solution. Our method is able to compute the first solution faster in all but one of the scenarios. This is shown in Table 5.2 in which we report mean first and best solution computation times for all Monte Carlo runs. Note also how RRT is the fastest method in computing its best solution, whereas RRT* is an order of magnitude slower. The reason is because, each reconnection step involves a sum of costly operations: computing Dubins paths, reprojection, and collision detection, which for RRT* is computed for all neighbors within a radius, a significantly larger number of times than in our approach, in which bookkeeping is performed.

Fig. 5.8 shows the computed paths for the six scenarios and all of the above-mentioned methods, but HRA*1. Notice that even when RRT* has smaller paths than HRA*, the paths computed by our method are smoother. Note also that the proposed approach cannot be as good as RRT* in finding the shortest path because it uses discretized configuration space for bookkeeping and rewiring, whereas RRT* explores the entire nearest neighbor set.

Finally, an interesting metric to compare path planning methods is the length vs time plot. This is, how quickly can any given method compute a solution with some given quality, say a fixed path length. These computational complexity per performance improvement plots are given in Fig. 5.9 for all the scenarios explored. The plots nicely show how RRT* is the most expensive algorithm able to compute shortest paths with computation times an order of magnitude larger than the rest of the methods. On the other side of the scale, RRT is the fastest of them all, but the path lengths it computes are in general larger than the rest of the methods. Furthermore, the plots also show that increased computational time does not necessarily mean better solutions with respect to path length for the case of RRT. Our proposed strategies HRA*1 and HRA*2, – randomized action sampling with heuristic cost penalties, with and without bookkeeping –, outperform competing methods in both ends. HRA* implementations take significantly less time to compute solutions with the same quality as those of RRT*, and are also able to compute shorter paths than RRT for the same allocated execution time.

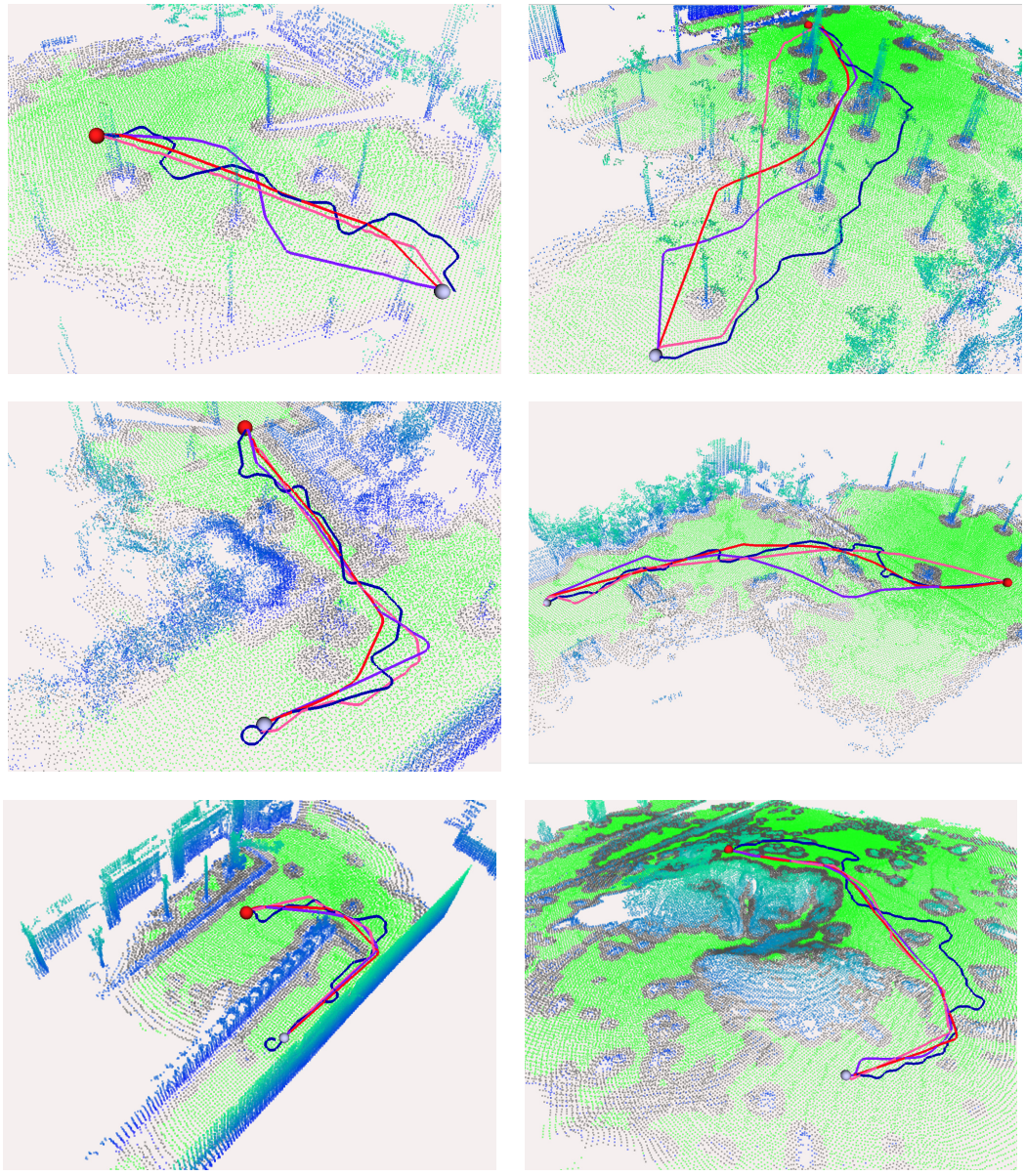


Figure 5.8: Path planning scenarios and computed paths. The scenarios, from upper left to bottom right, correspond to the FME1, FME2, FME3, FM4, BRL and BM subsets. The green dots indicate the traversable areas and the degraded blue and gray dots represent non-traversable regions. The start and goal positions are indicated by red and gray spheres, respectively. The resulting paths are: A* with rewiring (magenta), HRA* (red), RRT* (pink) and RRT (dark blue).

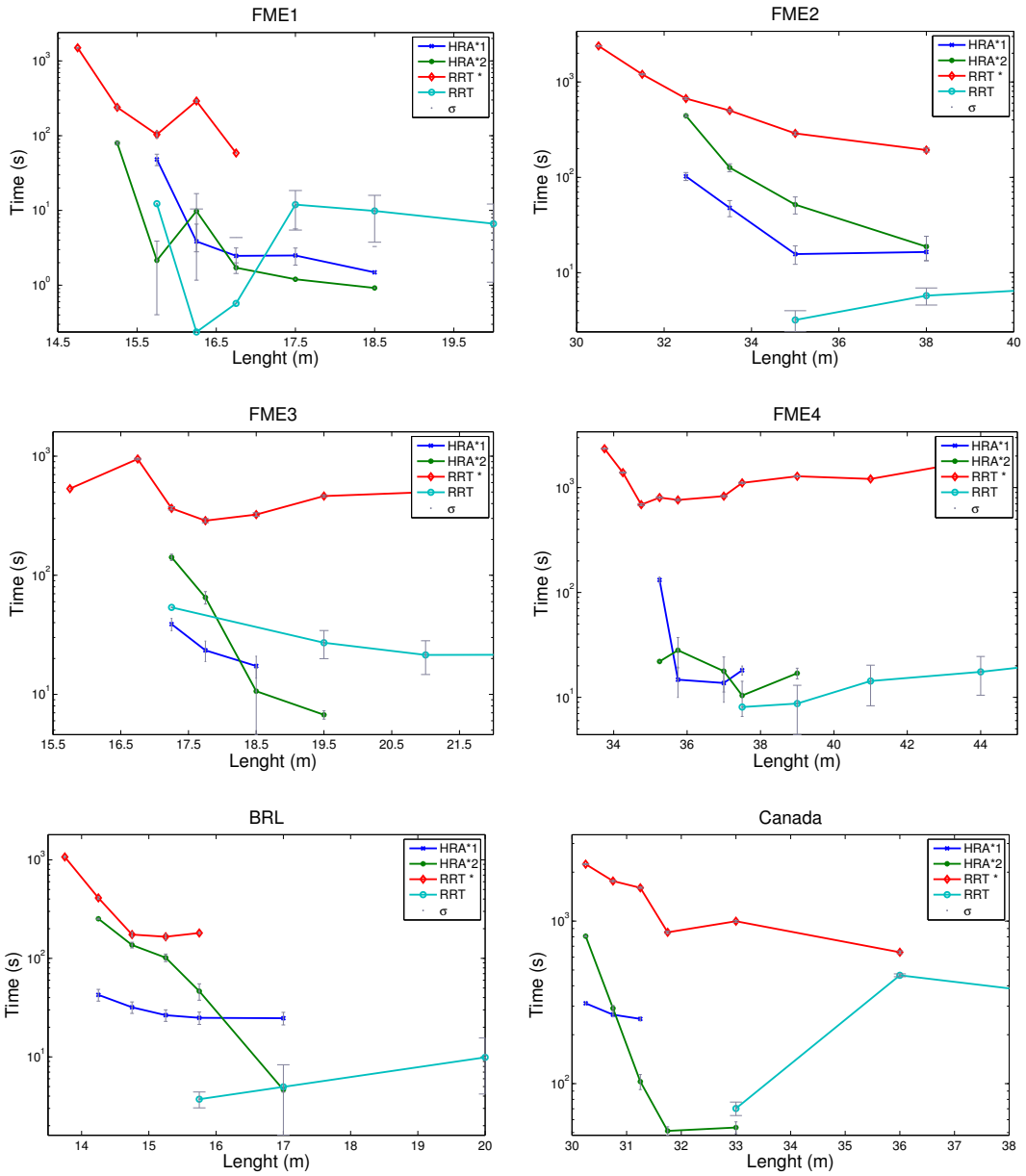


Figure 5.9: Length vs. time plots for the two HRA* proposed methods, RRT and RRT*. The grey ticks show one standard deviation bounds from the various Monte Carlo runs. Note the logarithmic scale of the time axis.

Chapter 6

Conclusions

Throughout this thesis we presented a framework for mapping, traversability analysis and planning for autonomous mobile robots in 3D environments using 3D range scanning as the main sensing device.

We summarize the key contributions as follows. We propose a hierarchical algorithm to compute the registration of point clouds, using point-to-point correspondence search at the coarser levels and point-to-plane correspondence search at the finest ones. Moreover, we propose a new very fast technique to compute multiple scan registration that benefits from prior known correspondences, as well as adequate linearized models of uncertainty for such registration estimates, which are used as inputs to a SLAM algorithm. These maps are then used for terrain classification, identifying safe navigable regions in the 3D map. These classified regions are further used in a proposed path planning scheme that computes feasible paths in 3D matching the kinematic constraints of the robot and which outperforms other state of the art planners.

In our study of the scan matching problem in **Chapter 2** we followed initially the spirit of the Iterative Closest Point algorithm [14], a common algorithm used for 2D point cloud registration in mobile robotics and computer vision [19, 43, 82], and further extended for 3D data [17, 95]. We proposed a new variation of the ICP, introducing a hierarchical new correspondence search that uses a combined point-to-plane and point-to-point correspondence search at different levels of granularity. The advantage of the point-to-plane projection is that it better models cloud fitting on sparse regions on the point cloud using local planarity assumptions [20]. The use of this hybrid approach

allows to perform good fitting at irregular surfaces such as trees or bushes, and also at regular ones such as the surface or on walls. It should be noted however, that this increase in precision comes at the expense of slightly higher computational costs, since the fitting local planar patches to the entire point cloud increases the time execution by a constant factor with respect to the point-to-point only strategy.

In our approach we adopted the minimization metric proposed in [17], which introduces a weight factor for rotations. In our implementation, the weighting parameter is set slightly higher than as reported previously, producing better registration than what can be obtained with an Euclidean distance only. The setting of the value of this parameter is very sensitive. Its use helps avoid overshooting and consequently divergence during the minimization step of the ICP filtering by enlarging artificially the distance between candidate matches for different orientations. An order of magnitude increase in this parameter to a range between 45 and 70 seemed to work well for our data sets, in contrast to the original work of Biota with values of in the order of 3 to 5.

We also bring in a novel ICP variation for simultaneous multiple scan registration that benefits from prior known correspondences. Speed up gain is substantial when compared with other methods. The method uses a voxel structure to efficiently search for correspondences to the first cloud in the set. The method was devised to search for loop closure after long sequences in open loop traverse.

In **Chapter 3** we approached the problem of computing an approximate to the covariance of the ICP registration. The method linearly propagates the noise on the estimation of the 3D point coordinates to that of the rototranslation between the two point clouds being registered. Since the ICP is an iterative optimization algorithm, the method makes use of the implicit theorem, which allows to compute the Jacobian of the optimization even when its form is not explicit. Once these covariance estimates are found, they can be used as input to our SLAM choice, the Pose SLAM algorithm. Pose SLAM presents advantages related to scalability and precision by reducing the state size and the number of loop closures, considering only non-redundant poses and informative links. Thus, linearization effects are delayed, maintaining the filter consistent for longer sessions [51]. Although, we proved experimentally that ICP covariance estimates using the point-to-pint cost function are useful for our global estimation machinery, and we did not find inconsistencies in the filter, *i.e* the poses map covariances were also used

in a path planner with uncertainty [132], a further refinement would be to compute a more accurate covariance estimate for the point-to-plane projection. This would further improve our uncertainty estimates but presumably also adding extra computation.

In **Chapter 4** we presented a system for 3D terrain classification in outdoor environments. The method is a high-level off-line terrain classification mechanism that processes 3D point clouds to generate traversability maps for the computation of global paths. The method uses Gaussian processes to classify the terrain as traversable or not, and has the advantage that it can be trained purely from positive samples. These samples can easily be acquired whilst maneuvering the robot in the intended terrain. Using two variants of supervised learning –GP regression and GP classification– we are able to classify dense point clouds acquired with Pose SLAM. We showed that with only two features, one for local roughness, and one for slope, we can get classification performance with f-scores better than SVM and naive parametric classification. Collision free regions for path planning were extracted from the filtered classified points, using obstacle detection and a novel border detection heuristic. Given that the GP encodes also the variance of the distribution, we leave also as future work, exploiting such information to also guide the path planning strategy, making the robot navigate along areas with least classification uncertainty, such as in [139].

Finally, we presented in **Chapter 5** a method to compute paths for a mobile robot in outdoor challenging environments. The method, called HRA*, is a modified A* algorithm that uses a hybrid node expansion technique that combines a random exploration of the action space meeting vehicle kinematic constraints, with a cost to goal metric that considers only kinematically feasible paths to the goal. The method is extended with a number of heuristics to penalize configurations near obstacles that accelerate search time. The technique was successfully tested on several real outdoors environments and was shown to outperform A* with rewiring, RRT and RRT* in computation time, and A* with rewiring and RRT in path length.

The three topics addressed, point cloud registration, terrain classification, and path planning, serve as a basis for a complete solution for the autonomous navigation of mobile robots in complex 3D terrains.

Bibliography

[1] D. Akca and A. Gruen. Fast correspondece search for 3D surface matching. *ISPRS Workshop Laser Scanner*, pages 186–191, September 2005.

[2] D. Akca and A. Gruen. Fast correspondece search for 3D surface matching. In *Proc. ISPRS Workshop on Laser Scanning*, pages 186–191, Enschede, Sep. 2005.

[3] J. Andrade-Cetto, M. Morta, P. Grosch, and E.H. Teniente. Dispositivo medidor de distancia y magnitudes omnidireccional. Patent application P201231044, Spanish Patent and Trademark Office, 2012.

[4] J. Andrade-Cetto, A. Ortega, E. Teniente, E. Trulls, R. Valencia, and A. Sanfeliu. Combination of distributed camera network and laser-based 3D mapping for urban service robotics. In *Proc. IEEE/RSJ IROS Workshop Network Robot Syst.*, pages 69–80, Saint Louis, Oct. 2009.

[5] J. Andrade-Cetto and M. Villamizar. Object recognition. In J. G. Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 1–28. John Wiley & Sons, New York, 2007.

[6] A. Angelova, L. Matthies, D.M. Helmick, and P. Perona. Fast terrain classification using variable-length representation for autonomous navigation. In *Proc. 21st IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–8, Minneapolis, Jun. 2007.

[7] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of Markov random fields for segmentation of 3D scan data. In *Proc. 19th IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 169–176, San Diego, Jun. 2005.

[8] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, Nov. 1998.

[9] S. Arya and D.M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. ACM SIAM Sym. Discrete Algorithms*, pages 271–280, Austin, Jan. 1993.

[10] J Barraquand and J-C Latombe. Nonholonomic multibody mobile robots: controllability and motion planning in the presence of obstacles. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 2328–2335, Sacramento, Apr. 1991.

[11] O. Bengtsson and A.-J. Baerveldt. Robot localization based on scan-matching estimating the covariance matrix for the IDC algorithm. *Robotics Auton. Syst.*, 44(1):29–40, 2003.

[12] R. Benjemaa and F. Schmitt. Fast global registration of 3D sampled surfaces using a multi-z-buffer technique. *Image Vis. Comput.*, 17:113–123, 1999.

[13] J.L. Bentley. K-d trees for semidynamic point sets. *Annual ACM Symposium Computational Geometry*, pages 187–197, 1990.

[14] P.J. Besl and N.D. McKay. A method for registration of 3D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, Feb. 1992.

[15] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, volume 3, pages 2743–2748, Las Vegas, Oct. 2003. IEEE.

[16] L. Biota. Algoritmos de scan matching basados en métricas para estimar el movimiento de robots que se desplazan en espacios tridimensionales. Technical report, Centro Politécnico Superior de Zaragoza, Jun. 2005.

[17] L. Biota, L. Montesano, J. Minguez, and F. Lamiraux. Toward a metric-based scan matching algorithm for displacement estimation in 3D workspaces. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 4330–4332, Orlando, May 2006.

[18] G. Blais and M.D. Levine. Registering multiview range data to create 3D computer objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):820–824, Aug. 1995.

[19] A. Censi. An accurate closed-form estimate of ICP's covariance. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 3167–3172, Rome, Apr. 2007.

[20] Y. Chen and G. Medioni. Object modeling by registration os multiples ranges images. In *Proc. IEEE Int. Conf. Robotics Autom.*, volume 3, pages 2724–2729, Sacramento, Apr. 1991.

[21] D. Chetverikov. Fast neighborhood search in planar point sets. *Pattern Recognit. Lett.*, 12(7):409–412, Jul. 1991.

[22] D. Chetverikov, D. Stepanov, and P. Krsek. Robust euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm. *Image and Vision Computing*, 23(3):299–309, 2005.

[23] D.M. Cole and P.M. Newman. 3D SLAM in outdoor environments. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1556–1563, Orlando, May 2006.

[24] A. Corominas Murtra, E. Trulls, J.M. Mirats-Tur, and A. Sanfeliu. Efficient use of 3D environment models for mobile robot simulation and localization. In *Proceedings of the International Conference on Simulation, Modelling and Programming for Autonomous Robots (SIMPAR'10), Lecture Notes on Artificial Intelligence*, Darmstadt, Germany, November 2010.

[25] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *J. ACM*, 32(3):505–536, Jul. 1985.

[26] G. Dissanayake, S. B. Williams, H. Durrant-Whyte, and T. Bailey. Map management for efficient simultaneous localization and mapping (SLAM). *Auton. Robots*, 12(3):267–286, May 2002.

[27] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robotics Res.*, 29(5):485–501, 2010.

[28] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3D lidar point clouds. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 2798–2805, Shanghai, May 2011.

[29] L.E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *Am. J. Math.*, 79:497–516, 1957.

[30] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[31] C. Elkan and K. Nato. Learning classifiers from only positive and unlabelled data. In *Proc. 14th Int. Conf. Knowl. Discov. Data Min.*, pages 213–220, Las Vegas, 2008.

[32] R. M. Eustice, H. Singh, and J. J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Trans. Robotics*, 22(6):1100–1114, Dec. 2006.

[33] O. Faugeras. *Three-Dimensional Computer Vision. A Geometric Viewpoint.* The MIT Press, Cambridge, 1993.

[34] D. Ferguson and A. Stentz. Anytime, dynamic planning in high-dimensional search spaces. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1310–1315, Rome, Apr. 2007. IEEE.

[35] S. Foix, G. Alenyà, J. Andrade-Cetto, and C. Torras. Object modeling using a ToF camera under an uncertainty reduction approach. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1306–1312, Anchorage, May 2010.

[36] E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *J. Guid. Control Dyn.*, 25(1):116–129, 2002.

[37] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM T. Math. Software*, 3(3):209–226, Sep. 1977.

[38] R. Galego, A. Ortega, R. Ferreira, A. Bernardino, J. Andrade-Cetto, and J. Gaspar. Uncertainty analysis of the DLT-lines calibration algorithm for cameras with radial distortion. *Comput. Vis. Image Underst.*, 140:115–126, 2105.

[39] D. Gingras, E. Dupuis, G. Payre, and J. de Lafontaine. Path planning based on fluid mechanics for mobile robots using unstructured terrain models. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1978–1984, Anchorage, May 2010.

[40] H. González-Baños and J.C. Latombe. Navigation strategies for exploring indoor environments. *Int. J. Robotics Res.*, 21(10):829–848, 2002.

[41] M. Greenspan and G. Godin. A nearest neighbor method for efficient ICP. In *Proc. 3rd Int. Conf. 3D Digital Imaging Modeling*, pages 161–168, Quebec, May 2001.

[42] S. Gumhold, X. Wang, and R. MacLeod. Feature extraction from point clouds. In *Proc. 10th Int. Meshing Roundtable*, pages 293–305, New Port Beach, Oct. 2001.

[43] J.-S. Gutmann. *Robuste Navigation autonomer mobile Systeme*. PhD thesis, University of Freiburg, Germany, 2000.

[44] M. Häselich, D. Lang, M. Arends, and D. Paulus. Terrain classification with Markov random fields on fused camera and 3D laser range data. In *Proc. Eur. Conf. Mobile Robots*, pages 153–158, Orebro, Sep. 2011.

[45] M. Hebert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 997–1002, Scottsdale, May 1989.

[46] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness. Towards learned traversability for robot navigation: From underfoot to the far field. *J. Field Robotics*, 23(11-12):1005–1017, 2006.

[47] A. Howard, D.F. Wolf, and G.S. Sukhatme. Towards 3D mapping in large urban environments. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 419–424, Sendai, Sep. 2004.

[48] S. Huang, Z. Wang, and G. Dissanayake. Exact state and covariance sub-matrix recovery for submap based sparse EIF SLAM algorithm. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1868–1873, Pasadena, May 2008.

[49] M. Hwangbo, J. Kuffner, and T. Kanade. Efficient two-phase 3D motion planning for small fixed-wing UAVs. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1035–1041, Rome, Apr. 2007.

[50] V. Ila, J. Andrade-Cetto, R. Valencia, and A. Sanfeliu. Vision-based loop closing for delayed state robot mapping. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3892–3897, San Diego, Nov. 2007.

[51] V. Ila, J. M. Porta, and J. Andrade-Cetto. Information-based compact Pose SLAM. *IEEE Trans. Robotics*, 26(1):78–93, Feb. 2010.

[52] V. Ila, J.M. Porta, and J. Andrade-Cetto. Reduced state representation in delayed state SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4919–4924, Saint Louis, Oct. 2009.

[53] V. Ila, J.M. Porta, and J. Andrade-Cetto. Amortized constant time state estimation in Pose SLAM and hierarchical SLAM using a mixed Kalman-information filter. *Robotics Auton. Syst.*, 59(5):310–318, 2011.

[54] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard. Autonomous exploration for 3D map learning. In Karsten Berns and Tobias Luksch, editors, *Autonome Mobile Systeme*, pages 22–28, Kaiserslautern, Oct. 2007. Springer.

[55] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, 2008.

[56] S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *Proc. IEEE Conf. Decision Control*, pages 7681–7687, Atlanta, Dec. 2010.

[57] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Res.*, 30(7):846–894, 2011.

[58] S. Karaman, M.R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1478–1483, Shanghai, May 2011.

[59] S. Karumanchi, T. Allen, T. Bailey, and S. Scheding. Non-parametric learning to aid path planning over slopes. *Int. J. Robotics Res.*, 29(8):997–1018, May 2010.

[60] L. Kavraki, M.N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Trans. Robotics Autom.*, 14(1):166–171, 1998.

[61] L. Kavraki, P. Svestkaand, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. Robotics*, 12(4):566–580, 1996.

[62] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 518–525, Orlando, May 2006.

[63] M. Kobilarov. Cross entropy motion planning. *Int. J. Robotics Res.*, 31(7):855–871, 2012.

[64] M. B. Kobilarov and G. S. Sukhatme. Near time-optimal constrained trajectory planning on outdoor terrain. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1821–1828, Barcelona, Apr. 2005.

[65] K. Konolige and M. Agrawal. FrameSLAM: from bundle adjustment to realtime visual mapping. *IEEE Trans. Robotics*, 24(5):1066–1077, 2008.

[66] D.J. Kriegman, E. Triendl, and T.O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Trans. Robotics Autom.*, 5(6):792–803, 1989.

[67] J. Kuffner and S. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. Robotics Autom.*, volume 2, pages 995–1001, San Francisco, Apr. 2000.

[68] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, JP How, and G. Fiore. Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Control Syst. Technol.*, 17(5):1105–1118, 2009.

[69] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo. Rover self localization in planetary-like environments. In *Proc. 3th Int. Sym. Artif. Intell., Robotics Autom. Space*, pages 433–440, Noordwijk, Jun. 1999.

[70] J. F. Lalonde, N. Vandapel, D. Huber, and M. Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *J. Field Robotics*, 23(1):839–861, Nov. 2006.

[71] P. Lamon, C. Stachniss, R. Triebel, P. Pfaff, C. Plagemann, G. Grisetti, S. Kolski, W. Burgard, and R. Siegwart. Mapping with an autonomous car. *In IEEE/RSJ IROS Workshop: Safe Navigation in Open and Dynamic Environments*, Oct. 2006.

[72] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic, London, 1991.

[73] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Comp. Sc. Dept., Iowa St. Univ., 1998.

[74] S. LaValle and J.Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 473–479, Detroit, May 1999.

[75] S. LaValle and J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics:New Directions*, 2000.

[76] S.M. LaValle. *Planning Algorithm*. Cambridge University Press, 2006.

[77] K. Lingemanna, A. Nüchter, J. Hertzberg, and H. Surmann. High-speed laser localization for mobile robots. *Robotics and Autonomous Systems*, 51(4):275–296, 2005.

[78] B. Liu, W.S. Lee, P.S. Yu, and X. Li. Partially supervised classification of text documents. In *Proc. 19th Int. Conf. Mach. Learning*, pages 387–394, Sydney, Jul. 2002.

[79] M.C. Martin and H.P. Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute. Carnegie Mellon University, 1996.

[80] T. Masuda. Registration and integration of multiple range images by matching signed distance fields for object shape modeling. *Comput. Vis. Image Underst.*, 87(1):51–65, 2002.

[81] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3D model construction. *International Conference on Pattern Recognition*, 20(1):879–883, 1996.

[82] J. Minguez, F. Lamiraux, and L. Montesano. Metric-based scan matching algorithms for mobile robot displacement estimation. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 3568–3574, Barcelona, Apr. 2005.

[83] J. Minguez, L. Montesano, and F. Lamiraux. Metric-based iterative closest point scan matching for sensor displacement estimation. *IEEE Trans. Robotics*, 22(5):1047–1054, Oct. 2006.

[84] H. P. Moravec. Robot spatial perception by stereoscopic vision and 3D evidence grids. Technical report, Carnegie Mellon University, The Robotics Institute, 1996.

[85] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 116–121, St. Louis, Mar. 1985.

[86] F. Moreno-Noguer, V. Lepetit, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *Int. J. Comput. Vis.*, 81(2):155–166, 2009.

[87] M. Morta. Disseny i construcció d'un laser 3D per al mapejat d'entorns exteriors. Technical report, Escola Técnica Superior d'Enginyería Industrial de Barcelona, June 2008.

[88] R.D. Morton and E. Olson. Positive and negative obstacle detection using the HLD classifier. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1579–1584, San Francisco, Sep. 2011.

[89] D. M. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. In *Proc. 2nd Fall Workshop Comput. Geom.*, Durham, Oct. 1997.

[90] D. Munoz, N. Vandapel, and M. Hebert. Onboard contextual classification of 3-D point clouds with learned high-order markov random fields. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 2009–2016, Kobe, May 2009.

[91] L. Murphy, S. Martin, and P. Corke. Creating and using probabilistic costmaps from vehicle experience. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4689–4694, Vilamoura, Oct. 2012.

[92] L. Murphy and P. Newman. Planning most-likely paths from overhead imagery. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 3059–3064, Anchorage, May 2010.

[93] D. Murray and C. Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proc. IEEE Int. Conf. Robotics Autom.*, volume 2, Albuquerque, Apr. 1997.

[94] D. Murray and J.J. Little. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171, 2000.

[95] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM with approximate data association. In *Proc. 12th Int. Conf. Advanced Robotics*, pages 242–249, Seattle, Jul. 2005.

[96] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. Heuristic-based laser scan matching for outdoor 6D SLAM. In Heidelberg Springer Berlin, editor, *KI 2005: Advances in Artificial Intelligence*, volume 3698, pages 304–319. SpringerLink, September 2005. Heuristic-based laser scan matching for outdoor 6D SLAM.

[97] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM-3D mapping outdoor environments. *J. Field Robotics*, 24(8-9):699–722, 2007.

[98] A. Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6D SLAM with an application in autonomous mine mapping. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1998–2003, New Orleans, Apr. 2004.

[99] S. T. O'Callaghan and F. Ramos. Gaussian process occupancy maps. *Int. J. Robotics Res.*, 31(1):42–62, 2012.

[100] A. Ortega, B. Dias, E. Teniente, A. Bernardino, J. Gaspar, and Juan Andrade-Cetto. Calibrating an outdoor distributed camera network using laser range finder data. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 303–308, Saint Louis, Oct. 2009.

[101] A. Ortega, M. Silva, E.H. Teniente, R. Ferreira, A. Bernardino, J. Gaspar, and J. Andrade-Cetto. Calibration of an outdoor distributed camera network with a 3D point cloud. *Sensors*, 14(8):13708–13729, 2014.

[102] S-Y. Park and M. Subbarao. A fast point-to-tangent plane technique for multi-view registration. In *Proc. 4th Int. Conf. 3D Digital Imaging Modeling*, pages 276–283, Banff, Oct. 2003.

[103] R. Paul, R. Triebel, D. Rus, and P. Newman. Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2404–2410, Vilamoura, Oct. 2012.

[104] F. Pauling, M. Bosse, and R. Zlot. Automatic segmentation of 3D laser point clouds by ellipsoidal region growing. In *Proc. Australasian Conf. Robotics Autom.*, Sydney, 2009.

[105] P. Pfaff and W. Burgard. An efficient extension of elevation maps for outdoor terrain mapping. In *Proc. 5th Int. Conf. Field and Service Robotics*, volume 25 of *Springer Tracts Adv. Robotics*, pages 195–206, Port Douglas, Jul. 2005. Springer.

[106] P. Pfaff, R. Triebel, C. Stachniss, P. Lamon, w.Burgard, and R. Siegwart. Towards mapping of cities. In *Proc. IEEE Int. Conf. Robotics Autom.*, Rome, Apr. 2007.

[107] J.C. Platt. *Probabilities for SV machines. Advances in large margin classifiers.* Advances in Large Margin Classifiers. MIT Press, 2000.

[108] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical recipes in C*, volume 2. Cambridge university press Cambridge, 1992.

[109] K. Pulli. Multiview registration for large data sets. In *Proc. 2nd Int. Conf. 3D Digital Imaging Modeling*, pages 160–168, Ottawa, Oct. 1999.

[110] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* Adaptive Computation and Machine Learning. The MIT Press, 2006.

[111] C.E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (GPML) toolbox. *J. Mach. Learning Res.*, 11:3011–3015, 2010.

[112] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Trans. Robotics Autom.*, 8(5):501–518, 1992.

[113] P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984.

[114] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3rd Int. Conf. 3D Digital Imaging Modeling*, pages 145–152, Quebec, May 2001.

[115] A. Sanfeliu and J. Andrade-Cetto. Ubiquitous networking robotics in urban settings. In *Proc. IEEE/RSJ IROS Workshop Network Robot Syst.*, pages 14–18, Beijing, Oct. 2006.

[116] A. Santamaria-Navarro, E.H. Teniente, M. Morta, and J. Andrade-Cetto. Terrain classification in complex three-dimensional outdoor environments. *J. Field Robotics*, 32(1):42–60, 2015.

[117] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Robotics: Science and Systems V*, Seattle, Jun. 2009.

[118] C. Siagian, C. Chang, and L. Itti. Autonomous mobile robot localization and navigation using a hierarchical map representation primarily guided by vision. *J. Field Robotics*, 31(3):408–440, 2014.

[119] D. Silver, J. A. Bagnell, and A. Stentz. Applied imitation learning for autonomous navigation in complex natural terrain. In *Proc. 7th Int. Conf. Field and Service Robotics*, volume 62 of *Springer Tracts Adv. Robotics*, pages 249–259, Cambridge, Jul. 2009. Springer.

[120] D. Silver, J. A. Bagnell, and A. Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *Int. J. Robotics Res.*, 29(12):1565–1592, 2010.

[121] T. Simeon and B. Dacre-Wright. A practical motion planner for all-terrain mobile robots. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1357–1363, Tokyo, Jul. 1993.

[122] D.A. Simon, M. Hebert, and T. Kanade. Real-time 3D pose estimation using a high-speed range sensor. In *Proc. IEEE Int. Conf. Robotics Autom.*, volume 3, pages 2235–2241, New Orleans, Apr. 2004.

[123] B. Sofman, E. Lin, J. A. Bagnell, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learning. In *Robotics: Science and Systems II*, Philadelphia, Aug. 2006.

[124] D. Stavens and S. Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proc. 22nd Conf. Uncertain. Artif. Intell.*, pages 13–16, Cambridge, Jul. 2006.

[125] T. Stoyanov, M. Magnusson, H. Andreasson, and A.J. Lilienthal. Path planning in 3D environments using the normal distributions transform. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3263–3268, Taipei, Oct. 2010.

[126] H Surmann, J. Huser, and J. Wehking. Path planning far a fuzzy controlled autonomous mobile robot. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, volume 3, pages 1660–1665, 1996.

[127] H. Surmann, A. Nuchter, and J. Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *Robotics Auton. Syst.*, 45(3-4):181–198, 2003.

[128] E.H. Teniente and J. Andrade-Cetto. Registration of 3D point clouds for urban robot mapping. Technical report, IRI, UPC, 2008.

[129] E.H. Teniente and J. Andrade-Cetto. FaMSA: Fast multi-scan alignment with partially known correspondences. In *Proc. Eur. Conf. Mobile Robots*, pages 139–144, Orebro, Sep. 2011.

[130] E.H. Teniente and J. Andrade-Cetto. HRA*: Hybrid randomized path planning for complex 3D environments. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1766–1771, Tokyo, Nov. 2013.

[131] E.H. Teniente, M. Morta, A. Ortega, E. Trulls, and J. Andrade-Cetto. Barcelona Robot Lab data set. [online] http://www.iri.upc.edu/research/webprojects/pau/datasets/BRL/php/dataset.php, 2011.

[132] E.H. Teniente, R. Valencia, and J. Andrade-Cetto. Dense outdoor 3D mapping and navigation with Pose SLAM. In *Proc. III Workshop de Robótica: Robótica Experimental*, pages 567–572, Seville, 2011.

[133] *OBJ* file format, –. http://local.wasp.uwa.edu.au/~pbourke/dataformats/obj/.

[134] C.H. Tong, D. Gingras, K. Larose, T.D. Barfoot, and E. Dupuis. The canadian planetary emulation terrain 3D mapping dataset. [online]http://asrl.utias.utoronto.ca/datasets/3dmap/, 2012.

[135] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2276–2282, Beijing, Oct. 2006.

[136] E. Trulls, A. Corominas Murtra, J. Pérez-Ibarz, G. Ferrer, D. Vasquez, Josep M. Mirats-Tur, and A. Sanfeliu. Autonomous navigation for mobile service robots in urban pedestrian environments. *Journal of Field Robotics*, 28(3):329–354, 2011.

[137] J. Tu and S. Yang. Genetic algorithm based path planning for a mobile robot. In *Proc. IEEE Int. Conf. Robotics Autom.*, volume 1, pages 1221–1226, Taipei, Sep. 2003.

[138] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Computer Graphics. Proc. ACM SIGGRAPH Conf.*, pages 311–318, Orlando, Jul. 1994. ACM Press.

[139] R. Valencia, M. Morta, J. Andrade-Cetto, and J.M. Porta. Planning reliable paths with Pose SLAM. *IEEE Trans. Robotics*, 29(4):1050–1059, 2013.

[140] R. Valencia, E.H. Teniente, E. Trulls, and J. Andrade-Cetto. 3D mapping for urban service robots. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3076–3081, Saint Louis, Oct. 2009.

[141] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte. Gaussian process modeling of large-scale terrain. *J. Field Robotics*, 26(10):812–840, 2009.

[142] T. Vidal-Calleja, A.J. Davison, J. Andrade-Cetto, and D.W. Murray. Active control for single camera SLAM. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1930–1936, Orlando, May 2006.

[143] C. Warren. Global path planning using artificial potential fields. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 316–321, Scottsdale, May 1989.

[144] J. Weingarten and R. Siegwart. EKF-based 3D SLAM for structured environment reconstruction. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3834–3839, Beijing, Oct. 2006.

[145] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner. Benchmarking urban six-degree-of-freedom simultaneous localization and mapping. *J. Field Robotics*, 25(3):148–163, Mar. 2008.

[146] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. IEEE ICRA Workshop Best Practices 3D Perception and Modeling for Mobile Manipulation*, Anchorage, May 2010.

[147] S.M. Yamany, M.N. Ahmed, and A.A. Farag. A new genetic-based technique for matching 3-D curves and surfaces. *Pattern Recognit.*, 32(10):1827–1820, 1999.

[148] B. Yamauchi. Frontier-based exploration using multiple robots. In *Int. Conf. Autonomous Agents*, pages 47–53, Minneapolis, 1998.

[149] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vis.*, 13:119–152, 1994.